WŁADYSŁAW TURSKI

PODSTAWY UŻYTKOWANIA MASZYN CYFROWYCH

PAŃSTWOWE WYDAWNICTWO NAUKOWE

WŁADYSŁAW TURSKI

PODSTAWY UŻYTKOWANIA MASZYN CYFROWYCH

W OŚRODKACH NAUKOWO-TECHNICZNYCH

WARSZAWA 1968
PAŃSTWOWE WYDAWNICTWO NAUKOWE

1

PRZEDMOWA

Adresatami tej książki są potencjalni użytkownicy sprzętu liczącego, a więc wszyscy aktywni pracownicy nauki i studenci kierunków matematyczno-przyrodniczych i technicznych, a także większość inżynierów. Ubolewam nad tym, że nie czułem się na siłach rozszerzyć tematyki książki o zagadnienia specyficzne dla zastosowań maszyn cyfrowych w gospodarce i naukach humanistycznych.

Książka adresowana do tak szerokiego grona nie może, siłą rzeczy, być monografią specjalistyczną; nie było jednak moim celem napisanie li tylko pracy popularnej. Dlatego też zakładam, że czytelnik wie co to jest maszyna cyfrowa w tradycyjnym sensie i na czym polega jej programowanie. Książka M. Greniewskiego [40] zawiera wszelkie niezbędne w tym zakresie wiadomości i może być śmiało zalecana jako lektura uzupełniająca, a jej rozdziały 1 i 2 jako lektura wstępna. Zakładam także, że czytelnik posiada pewne wiadomości o programowaniu w językach algorytmicznych, a zwłaszcza w Algolu. Literaturę uzupełniającą w tym względzie stanowić może monografia S. Paszkowskiego [86].

Przy pisaniu korzystałem z licznych uwag i komentarzy prywatnych i seminaryjnych mych kolegów z Centrum Obliczeniowego PAN (gdzie książka została napisana) i Sekcji Maszyn Matematycznych Imperial College of Science and Technology (gdzie zrodził się pomysł jej napisania). Kierownikom obydwu placówek, profesorom M. Warmusowi i S. Gillowi składam niniejszym serdeczne podziękowania za wszelką okazaną mi pomoc.

Dziękuję również wszystkim osobom i instytucjom, które udostępniły mi szereg niepublikowanych materiałów, a w szczególności firmom California Computer Products, Inc., Facit AB, International Business Machines, Inc., International Computers and Tabulators, Ltd. za materiały fotograficzne reprodukowane w sekcji 5.1.

Składam także serdeczne podziękowanie panu B. Randellowi za zezwolenie wykorzystania wielu przykładów zamieszczonych w sekcji 3.3.1, panu B. Svejgaardowi za udostępnienie niepublikowanej procedury wykorzystanej w Dodatku B oraz panu D. Holsteinowi za fotografie zamieszczone w sekcji 5.1.

Część rękopisu przejrzała pani magister E. Stolarska, której zobowiązany jestem za szereg pomocnych uwag natury redakcyjnej.

WŁADYSŁAW TURSKI

Centrum Obliczeniowe PAN, kwiecień 1967

2

WSTĘP

Trzeba z żywymi naprzód iść
Po życie sięgać nowe
A nie w uwiędłych laurów liść
Z uporem stroić głowę

(A. ASNYK)

Wedle początkowych zamiarów autora, książka ta miała nosić odmienny tytuł i obejmować szerszy krąg problemów: miała to być książka o metodach wykorzystania maszyn cyfrowych w pracy naukowej. W miarę gromadzenia materiałów i notatek okazało się jednak, ponad wszelką wątpliwość, że wstępne projekty obejmują zbyt wielką dziedzinę i trzeba je odpowiednio ograniczyć, tak by pozostały materiał mógł stanowić książkę o rozsądnych rozmiarach.

Decyzja ograniczenia treści nie była dla autora łatwa. Z jednej bowiem strony pragnął on przedstawić możliwie szerokim kręgom potencjalnych użytkowników współczesny dorobek metodologiczny w zakresie treści procesu przetwarzania informacji, z drugiej jednak strony był świadom kompletnego braku polskich opracowań z zakresu współczesnych poglądów na formę i organizację tego procesu oraz na ich zależność od struktury sprzętu liczącego. Treść książki, którą oddajemy w ręce czytelnika, jest odzwierciedleniem wniosku, do którego doszedł autor w wyniku własnych przemyśleń i rozmów z kolegami. Wniosek ten daje się wyrazić jak następuje:

Ze względu na istniejące opóźnienie w dziedzinie stosowania techniki cyfrowej w naszym życiu gospodarczym i naukowym oraz na realizowaną do niedawna strategię pełnego wzorowania się na „wypróbowanych", tj. w praktyce przestarzałych osiągnięciach zagranicznych, odbiciem której to strategii był także dobór publikowanych po polsku materiałów przeglądowych i podręczników, a także ze względu na burzliwy rozwój techniki użytkowania sprzętu liczącego po roku 1960, najważniejszym problemem w dziedzinie stosowania maszyn cyfrowych w naszym Kraju jest przygotowanie przyszłych ich użytkowników na przyjęcie rozwiązań nowoczesnych, odbiegających dosyć poważnie od rozwiązań stosowanych dotychczas. Jednym z pierwszorzędnych zadań w tym zakresie jest przekazanie informacji o aktualnym stanie wiedzy i tendencjach rozwojowych światowych badań nad metodami użytkowania sprzętu liczącego. W miarę swych sił i zdolności, autor niniejszej książki pragnął ten właśnie cel osiągnąć.

Wychodząc z takiego założenia co do treści i celu książki, nie trudno przewidzieć, że jest ona w dużym stopniu kompilacją, wychodząc zaś z naturalnego założenia ograni-

czenia objętości – że nie jest kompilacją encyklopedyczną. Jest więc kompilacją selektywną. Autor nie ma zamiaru łudzić czytelnika – selektywność kompilacji w tej książce oznacza jej tendencyjność. Autor nie ma zamiaru ukrywać przed czytelnikiem własnej fascynacji ogromnymi możliwościami nowoczesnej organizacji sprzętu liczącego i wspaniałymi osiągnięciami pionierskich nieraz prac w dziedzinie jej wykorzystania. Autor przyznaje, że książka ta jest odbiciem jego głębokiego przekonania o tym, że najbardziej owocnym podejściem do zagadnień sposobu użytkowania sprzętu liczącego jest zasada, iż całokształt procesu przetwarzania informacji winien być maksymalnie przystosowany do nawyków i tradycji pracy umysłowej człowieka, że celem procesu przetwarzania informacji jest przede wszystkim wykonanie określonych usług, a potem dopiero optymalizacja wykorzystania sprzętu.

Książka rozpada się na trzy duże działy:

Dział pierwszy odpowiada rozdziałowi 3 i poświęcony jest opisowi niektórych języków programowania automatycznego i podstawowym wiadomościom z zakresu budowy translatorów. Dobór języków omówionych w książce nie jest przypadkowy. Pierwsze dwa stanowią najważniejsze obecnie języki uniwersalne do zastosowań naukowych. Fortran został opisany nieco obszerniej i z pewnym nastawieniem dydaktycznym, ponieważ brak jest w języku polskim odpowiedniego materiału źródłowego. Przy okazji omawiania Algolu pozwolił sobie autor na odejście od czysto referatowego charakteru książki; być może zamieszczone tam uwagi skłonią czytelnika do głębszych refleksji.

Języki LISP i SOL stanowią przykłady popularnych języków „obsługujących" ważne i dosyć specjalne kręgi problemów. Włącznie ich do naszego tekstu winno zorientować czytelnika w bogactwie i różnorodności stosowanych języków. Krótka sekcja o PL/I nosi charakter informacyjny i zwraca uwagę czytelnika na język, o którym będzie on słyszał bardzo wiele w przyszłości.

Podstawowe wiadomości o budowie translatorów i różnych metodach podejścia do tej problematyki mają wypełnić dwie funkcje: po pierwsze – zaspokoić naturalną ciekawość człowieka, który chce wiedzić jak zbudowane jest narzędzie, którym się posługuje, i po drugie – uzbroić tych spośród czytelników, którzy chcieliby sami budować translatory zaprojektowanych przez siebie języków specjalistycznych, w elementarne wiadomości o metodach tej pracy. Doświadczenie wskazuje bowiem, że często brak tych właśnie elementarnych wiadomości stoi na przeszkodzie praktycznej realizacji ciekawych skąd innąd pomysłów.

Dział drugi odpowiada rozdziałowi 4 i poświęcony jest współczesnej architekturze systemów liczących.

Autor wyraża nadzieję, że dział ten poinformuje czytelnika o przyczynach, które spowodowały tak znaczną różnorodność trybów eksploatacji sytemów liczących, jaką obserwujemy obecnie i zaznajomi go z podstawowymi środkami ich realizacji. Celem autora jest ukazanie dynamizmu rozwoju poglądów na architekturę systemów liczących i udokumentowanie tezy, że przyjęcie odpowiedniej struktury podstawowego oprogramowania systemu może, przy stosunkowo niewielkich zmianych w samym sprzęcie, pozwolić na osiągnięcie trybu eksploatacji odpowiadającego indywidualnym potrzebom użytkowników.

Jest także niezłomnym przekonaniem autora, że powszechna (wśród potencjalnych użytkowników) świadomość możliwości wyboru nie tylko sprzętu i jego konfiguracji, ale także trybu eksploatacji stanowi najważniejszy (spośród wielu) warunek racjonalnego stosowania techniki cyfrowej.

Ostatni dział obejmuje rozdziały 5 i 6. Dział ten zawiera rozważania na temat wymiany informacji pomiędzy człowiekiem i systemem liczącym, tak pod względem urządzeń, które wymianę tę umożliwiają, jak i pod względem trybu, w którym się ona odbywa, oraz kilka uwag, które mogą być pomocne przy podejmowaniu decyzji leżących u podstaw organizacji nowych ośrodków obliczeniowych.

Obserwując działalność istniejących ośrodków obliczeniowych w Kraju, autor napotkał trudny do wyjaśnienia fenomen skrajnego ubóstwa wyposażenia w środki wymiany informacji, zwłaszcza w dziedzinie urządzeń graficznych. Nie wydaje się przy tym, by względy finansowe odgrywały tutaj najważniejszą rolę, gdyż koszt powszechnie stosowanych urządzeń perforujących najwyższej jakości jest porównywalny z kosztem mechanicznych urządzeń graficznych. Nasuwa się więc przypuszczenie, że również pod tym względem brak informacji wśród szerokich kręgów użytkowników jest główną przyczyną.

Uwagi o organizacji ośrodka, zawarte w rozdziale 6 okażą się zapewne dla wielu czytelników zupełnie oczywiste. Autor nie ma zamiaru polemizować z takim stwierdzeniem, co więcej, byłby bardzo szczęśliwy, gdyby dla wszystkich organizatorów przyszłych i kierowników obecnych ośrodków uwagi i sugestie z tego rozdziału okazały się dobrze znanymi faktami.

Poszczególne działy są w dużym stopniu wzajemnie niezależne. Rozdziały 3 i 4 mogą być czytane osobno i w dowolnej kolejności. Rozdziały końcowe lepiej jest poprzedzić lekturą pierwszych dwu działów.

Uzupełnienie książki stanowią dodatki, z tekstu książki prowadzą do nich odsyłacze; większość z nich może być jednak traktowana jako zamknięte i samodzielne całości.

Obficie cytowana literatura zebrana została w porządku alfabetycznym, by formą zgrupowania nawet nie pretendować do miana bibliografii. Nie jest ona bowiem ani zupełna, ani systematyczna. Obejmuje ona jednak kilka pozycji, które autor gorąco zaleca zaiteresowanym czytelnikom jako materiał uzupełniający do niniejszej książki; odpowiednie wskazówki zawarte zostały w tekście.

Na zakończenie winien jest autor kilka słów wyjaśnienia i przeprosin pod adresem czytelnika. Czytelnik zechce mianowicie wybaczyć, że tekst książki nie jest opracowany dostatecznie systematycznie pod względem terminologicznym, nie wykluczone są również pewne sprzeczności i rozbieżności terminologiczne w obrębie samej książki (chociaż autor dokładał wielu starań, by ich uniknąć przynajmniej w obrębie poszczególnych sekcji). Przyczyna jest jedna – polska terminologia przedmiotu książki znajduje się faktycznie dopiero *in statu nascendi*. Oczywiście, przy pewnym nakładzie pracy można by było potraktować niniejszą książkę także jako próbę ustalenia polskiej terminologii. Autor nie wybrał tej drogi z dwu powodów: po pierwsze, nie ma on żadnych ambicji w tym kierunku, po drugie zaś, przedłużyło by to proces przygotowania książki do druku, a więc przeczyło pragnieniu możliwie małego opóźnienia informacji zawartej w książce w stosunku do informacji źródłowej.

3

JĘZYKI PROGRAMOWANIA

3.1. ROZWÓJ, KLASYFIKACJA I EFEKTYWNOŚĆ

Zasadniczym celem wykorzystania maszyn matematycznych jest zwiększenie możliwości przetwarzania informacji zarówno pod względem ilości jak i szybkości oraz ułatwienie samego procesu przetwarzania. Pierwsze dwa aspekty uwarunkowane są osiągnięciami technicznymi, aspekt ostatni zależy od stopnia zautomatyzowania procesu programowania i obsługi maszyn cyfrowych. Rozwój technologii budowy maszyn cyfrowych w ostatnich latach doprowadził do tego, że problemem najważniejszym, kluczem dalszego rozwoju zastosowań stała się automatyzacja programowania i obsługi maszyn do przetwarzania informacji.

Rozdział niniejszy poświęcimy krótkiemu przeglądowi kilku ciekawszych języków automatycznego programowania, stanowiących w pewnym sensie najpopularniejsze przykłady wyników dociekań nad sposobami automatyzacji programowania. Języki programowania, o których będziemy mówili w sekcji 3.2 i metody budowy translatorów, którym poświęcona jest sekcja 3.3, nie stanowią jednak jedynych form automatyzacji programowania. Zarówno historycznie jak i logicznie poprzedzone są one znacznie prostszymi systemami, o których pomówimy nieco niżej. Trzeba także zwrócić uwagę czytelnika na fakt, że najbardziej nawet rozbudowany język programowania wraz ze swoim translatorem nie wyczerpuje wszystkich możliwości automatyzacji programowania, a zwłaszcza obsługi maszyn matematycznych (por. rozdział 4 i sekcję 5.1).

Notka historyczna

Idea automatyzacji programowania jest równie stara jak idea programowania maszyny liczącej. Prekursorką nowoczesnych maszyn liczących była (niezrealizowana zresztą do końca) Maszyna Analityczna Charlesa Babbage'a. W roku 1842 F. L. Menbrea pisał([1]):

([1]) Cytat za angielskim przekładem, dokonanym przez Adę Augustę, hrabinę Lovelace, córkę Byrona, przytoczonym w przedmowie napisanej przez H. D. Huskeya do książki Halsteada [42]. N. B., sama Ada Augusta przewidywała zastosowanie Maszyny Analitycznej do symbolicznego przetwarzania informacji.

„Skoro więc Maszyna będzie skonstruowana, trudności [wykonywania obliczeń] sprowadzą się do przygotowania kart [perforowanych], lecz jako że te ostatnie są zwykłym przekładem wzorów algebraicznych, nietrudno będzie używając *odpowiednio prostą notację* (podkreślenie moje — W. T.) zlecić przygotowanie kart pracownikom niewykwalifikowanym. Tak więc cały wysiłek intelektualny ograniczy się do przygotowania wzorów, które muszą być przystosowane do wyliczeń przez Maszynę".

Historia współczesnych maszyn liczących rozpoczęła sie dnia 7 sierpnia 1944 roku, gdy ruszył Automatyczny Kalkulator Uniwersytetu Harvarda. W niedługim czasie zakończone zostały prace nad pierwszą maszyną opartą na technice elektronicznej: ENIAC. Na przełomie lat czterdziestych i pięćdziesiątych elektroniczne maszyny cyfrowe produkowane już były w trzech krajach: USA, Wielkiej Brytanii i ZSSR. W miarę wzrostu ilości maszyn cyfrowych i ich zastosowań coraz bardziej dawały się odczuć trudności związane z „naturalnym systemem programowania" w języku binarnym i jego reprezentacjach przy innych podstawach rozwinięć systematycznych: 8, 10, 16. Programowanie takie powodowało powstawanie „wąskich gardeł" w procesie przetwarzania informacji, którym zresztą w owych czasach było z reguły numeryczne przetwarzanie danych numerycznych. Nieadekwatność takiego programowania do rysujących się możliwości zastosowania maszyn cyfrowych uwypukliła się tym bardziej, gdy w roku 1947 J. v. Neuman wykazał, iż przy odpowiednim formacie słowa maszyny zanika potrzeba różnicowania ośrodków przechowujących (pamięci) instrukcje i dane, a co za tym idzie powstaje możliwość operowania na instrukcjach. Równocześnie w Wielkiej Brytanii prowadzone były pod kierownictwem A. M. Turinga w Narodowym Laboratorium Fizycznym prace nad Automatyczną Maszyną Liczącą (ACE — *Automatic Computing Engine*) i nad językami przydatnymi do formułowania zadań dla maszyn liczących. Prace te doprowadziły do powstania i zastosowania pojęcia rozszerzonej instrukcji. Pojęcie to odżyło w ostatnich latach pod nieco zmienioną nazwą *makrorozkazu,* albo po prostu: *makro*[1]. Makrorozkaz jest instrukcją przyjmującą postać rozkazu maszynowego; w odróżnieniu od zwykłego rozkazu nie jest on jednak wykonywany bezpośrednio, lecz powoduje wykonanie określonego ciągu rozkazów. Koncept makrorozkazów jest niezwykle płodną postacią automatyzacji programowania; niektóre konsekwencje makroprogramowania omówimy w sekcji 3.3.2. Do szkoły Turinga należy także priorytet wprowadzenia zapisu funkcyjnego dla oznaczenia czynności wykonywanych przez maszynę.

W pierwszej połowie lat pięćdziesiątycc następuje rozwój burzliwej automatyeacji programowania. Najpowszechniejszą formą programowania automatycznego jest wówczas *programowanie symboliczne.* W przeciwieństwie do badań Turinga, noszących charakter teoretyczny, programowanie symboliczne wywodzi się z przesłanek czysto pragmatycznych. Aby umożliwić wykorzystanie programu realizującego pewne czynności przy rozwiązywaniu różnych zadań, w skład których to rozwiązań czynności te wchodzą, należało zrezygnować z założenia, że dany program zajmuje ustalone miejsce w pamięci maszyny. Przyjęto zamiast tego zasadę, że adresy występujące w programie są względne i muszą być przeliczane na adresy absolutne przed wykonaniem programu. Mamy więc do czy-

---

(1) Inne równoważne nazwy: rozkaz programowany, rozkaz syntetyczny.

nienia z prymitywnym przykładem translacji polegającym na prostym zastąpieniu adresów wyrażonych symbolicznie (zapis bywał tak cyfrowy jak i literowy, jednakże dalszy rozwój wykazał większą płodność zapisu literowego) przez adresy absolutne, stosując odpowiednie tablice lub proste reguły. Skoro jednak adresy podlegają translacji, to nic nie stoi na przeszkodzie by także i części operacyjne instrukcji były tłumaczone. Tak więc zamiast kodowania cyfrowego mamy już do czynienia z kodowaniem symbolicznym o znacznie większych walorach mnemotechnicznych: zamiast np. cyfrowego kodu operacji dodawania piszemy ADD (=dodaj), zamiast absolutnego adresu wielkości dodawanej piszemy symboliczną nazwę tego adresu, np. *X*.

Klasycznymi przykładami systemów programowania tego okresu są wyniki otrzymane przez grupę specjalistów z Carnegie Institute of Technology, kierowaną przez A. J. Perlisa — język IT i opracowany przez koncern IBM język SOAP.

Zjawiskiem charakterystycznym dla dalszego rozwoju automatyzacji programowania jest wzbogacenie języka o dalsze elementy werbalne — np. użycie słowa JUMP (=skocz) dla oznaczenia instrukcji zmiany sekwencji wykonywania instrukcji, a także wprowadzenie symboli operacji arytmetycznych z jednoczesnym zerwaniem z dotychczasowym układem leksykograficznym(1) programu: zamiast rozbicia na wiersze odpowiadające jednej instrukcji kodu maszyny pojawiają się wiersze reprezentujące pewne logiczne całości, np. wyrażenia algebraiczne. W Stanych Zjednoczonych typowymi przykładami są języki UNICODE, opracowany przez koncern Sperry Rand i elementarny Fortran, opublikowany w roku 1957 przez grupę współpracowników IBM, kierowaną przez J. W. Backusa. Warto zauważyć, że już w tym okresie zarysowuje się tendencja do pewnego ujednolicenia (z praktycznego punktu widzenia) systemów programowania, przejawiająca się np. w powstaniu systemu FORTRANSIT(2). System ten ma tak wielkie znaczenie historyczne, że zatrzymamy się nieco dłużej na opisie jego zasadniczej funkcji.

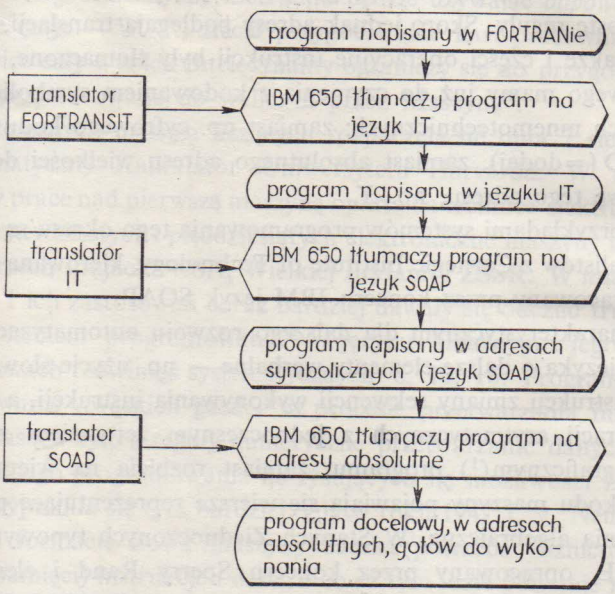
Język Fortran zrealizowany został po raz pierwszy na maszynie IBM 704, która w latach 1957/58 zastępowała najpopularniejszą uprzednio maszynę uniwersytecką IBM 650, wyposażoną często w translator języka IT. System FORTRANSIT zaprojektowany

---

(1) W tej książce używać będziemy często przymiotnika „leksykograficzny" w sensie nie mającym nic wspólnego z układaniem słowników, na co mogło by wskazywać encyklopedyczne wyjaśnienie rzeczownika „leksykografia". Przymiotnik ten, dla którego nie znaleźliśmy lepszego polskiego odpowiednika, używany będzie dla wyrażenia zespołu cech istotnych układu zapisu programu, a w szczególności: następstwa symboli, ich zgrupowania w konstrukcje właściwe dla konkretnego języka programowania i rozgraniczeń graficznych między poszczególnymi konstrukcjami. Mówiąc np. o następstwie leksykograficznym symboli mamy na myśli kolejność wynikającą z zapisu, a nie z wykonania programu. Dodać należy, że nie wszystkie cechy graficzne zapisu programu są cechami leksykograficznymi — np. przejście do nowego wiersza jest w języku Fortran cechą istotną i nie ma żadnego znaczenia przy zapisie w języku Algol 60. Można powiedzieć w przybliżeniu, że cechy leksykograficzne są graficznym odpowiednikiem składni, a jednostki leksykograficzne — jednostek syntaktycznych języka programowania. Użycie terminu „leksykograficzny" jest dość powszechne w tym znaczeniu w literaturze anglosaskiej, jest on również stosowany w niektórych polskich środowiskach obliczeniowych.

(2) Szczegółowy opis języków IT, SOAP, UNICODE i FORTRANSIT znaleźć można w wydanym przez A. P. Jerszowa zbiorze przekładów [54].

został w celu umożliwienia wykonywania programów napisanych w Fortranie na maszynie IBM 650. Schemat translacji wyglądał jak następuje:



Tego rodzaju wieloetapowe tłumaczenie programów nosi nazwę techniki *sznurowania* (*bootstrapping*); pomimo pozorów nadmiernego skomplikowania, metoda ta uprawiana jest po dzień dzisiejszy przy oprogramowaniu różnych maszyn oraz przy budowie translatorów z wykorzystaniem języków i translatorów poprzedniej generacji (por. np. [42]).

Spośród niewspominanych dotychczas ośrodków opracowujących systemy automatyzacji programowania wypada zwrócić uwagę na tzw. szkołę manchesterską (R. A. Brooker, patrz [6]) ściśle związaną z firmą Ferranti (S. Gill i inni, por. np. [37]). Do szczególnie poważnych osiągnięć tej grupy zaliczyć należy przede wszystkim opracowanie metod wykorzystania wielopoziomowej pamięci w ramach języków automatycznego programowania, metody te stosowane są aktywnie do dzisiaj. Inną dosyć charakterystyczną cechą języków opracowanych przez szkołę manchesterską jest dążenie do zapewnienia programiście możliwie pełnej kontroli nad wszystkimi elementami maszyny, tj. zachowanie w systemie automatycznego programowania możliwie wielu dodatnich aspektów programowania w języku zbliżonym do kodu wewnętrznego maszyny. Przykładem konkretnego zastosowania dwu wspomnianych charakterystyk szkoły manchesterskiej może służyć mechanizm segmentacji programów w języku *Mercury Autocode*, tj. mechanizm, dzięki któremu programista może dzielić program na mniejsze jednostki leksykograficzne, noszące nazwę rozdziałów (*chapters*), kierując się przy tem znajomością pojemności pamięci szybkiej, ilości miejsc roboczych, ilości funkcji standardowych etc(1).

(1) Podobny mechanizm zastosowany został w wielu językach programowania, np. w systemie KLIPA, opracowanym w Centrum Obliczeniowym PAN [23].

Nieco odmienny kierunek przyjęły wczesne prace nad automatyzacją programowania w ZSRR([1]). W oparciu o prace A. A. Liapunowa przedstawiające program wykonania obliczeń w postaci uporządkowanego ciągu operatorów rozwinęła się tam teoria tzw. *programowania operatorowego*, dla której językiem opisującym konkretne algorytmy stał się *język schematów logicznych* opracowany przez J. I. Janowa. Proces przygotowania programu rozpada się przy programowaniu operatorowym na szereg etapów, wypełnianych częściowo przez ludzi – np. sporządzenie schematu logicznego algorytmu, zaopatrzenie go w dodatkowe informacje wyjaśniające semantyczne znaczenie poszczególnych operatorów, a częściowo przez maszynę – np. zastąpienie adresów umownych przez konkretne. Charakterystycznym jest przy tym to, że nie tylko adresy ale i same operatory przedstawione są w postaci liczb umownych; tablica odpowiedniości operatorów i adresów umownych kodom i makrorozkazom oraz adresom absolutnym sporządzana jest także „ręcznie". Podejście takie uwarunkowane było w dużym stopniu brakiem wyposażenia maszyn radzieckich w urządzenia wejścia/wyjścia dopuszczające znaki alfabetyczne. Konieczność kodowania całych programów w postaci liczb umownych nie pozbawiona była jednak cech dodatnich – usunięte zostały np. ograniczenia na ilość dopuszczalnych symboli, jako że dowolny symbol można przedstawić w postaci liczby umownej. Dalszy rozwój programowania operatorowego i języka schematów logicznych doprowadził do powstania teorii równoważnych przekształceń programów i do rozwiązania innych ciekawych problemów teoretycznych, nie mających jednak, jak dotąd, poważniejszych zastosowań w praktyce.

Dalszy rozwój automatyzacji programowania rozpatrywać należy w kontekście różnych potrzeb, których zaspokojeniu miał służyć. Wspomnieliśmy na wstępie, że jedną z zasadniczych potrzeb było usprawnienie procesu programowania maszyn cyfrowych. Automatyzacja programowania oznacza w tym aspekcie działalność dwustronną: tworzenie języków programowania, mniej lub bardziej adekwatnych do mniej lub bardziej precyzyjnie określonych kręgów problemów i tworzenie programów umożliwiających translację i wykonanie programów napisanych w tych językach. Poniżej spróbujemy usystematyzować pewne informacje o tym aspekcie automatyzacji programowania. Drugą z potrzeb, które winny być zaspokojone przez automatyzację programowania jest cały skomplikowany zespół problemów związanych z procesem uruchamiania programów, o którym pomówimy w sekcji 5.2. Trzecią wreszcie dziedziną automatyzacji programowania stały się badania nad efektywnym wykorzystaniem nowoczesnych systemów przetwarzania informacji; a więc zadania związane z wykorzystaniem wieloprogramowanych, wielodostępnych i wielokrotnych konfiguracji maszyn cyfrowych. O niektórych rezultatach tych badań mówić będziemy w sekcji 4.2.

Zanim zamkniemy niniejszą notatkę historyczną zwrócimy jeszcze tylko uwagę na ciekawą tendencję zapoczątkowaną na dużą skalę przez Perlisa i Samelsona w roku 1957. Jest to tendencja do standaryzacji języków programowania drogą rozważań teoretycznych. Szeroki oddźwięk z jakim spotkały się propozycje tych dwu uczonych stworzył odpowied-

---

([1]) Przegląd literatury na ten temat zawiera praca W. Turskiego [103]; por. także książki: Kitowa i Krinickiego [64] oraz Krinickiego, Mironowa i Frołowa [65].

nie warunki do zwołania w maju 1958 r. w Zurichu międzynarodowej konferencji poświęconej opracowaniu standardowego języka algebraicznego. Język taki, znany pod nazwą Algol-58 został opracowany i poddany pod dyskusję zainteresowanych środowisk. Znacznie ulepszona wersja tego języka opracowana została pod auspicjami IFIP na konferencji w Paryżu w 1960 roku (Algol-60). Dalsza praca w tym kierunku doprowadziła do wydania skorygowanej wersji Algolu-60 (Rzym, kwiecień 1962). Biorąc pod uwagę rozwijające się pole zastosowań naukowych maszyn cyfrowych, IFIP uznała istniejący język Algol-60 za niewystarczający i powołała specjalną komisję dla opracowania nowego wariantu języka Algol.

Przytoczona krótka historia języka Algol – jedynego jak dotąd międzynarodowego standardu programowania – wymownie świadczy o trudności ustalenia w chwili obecnej kanonicznego języka programowania, który mógłby swą rolę spełniać na przestrzeni kilku chociażby lat. Przyczyną tego stanu rzeczy jest prosty fakt, że nasza wiedza o zakresie stosowalności maszyn cyfrowych jest w stanie gwałtownej ekspansji. Ezoteryczne badania dnia wczorajszego stają się chlebem powszednim dnia dzisiejszego, techniki programowania uważane za nowatorskie dzisiaj stają się przykładami podręcznikowymi jutro. W takiej sytuacji próby ustalenia kanonu o nieprzemijającej ważności czasowej są z góry skazane na niepowodzenie. Z drugiej jednak strony wzrasta rola standardowego języka programowania o znaczeniu czasowo lokalnym, języka, który stanowiłby układ odniesienia dla innych języków i umożliwiał wymianę informacji między ludźmi i maszynami w jakimś konkretnym interwale czasowym. Taką rolę w dużej mierze spełnił Algol-60.

Jeśli wolno pokusić się o próbę spekulacji, można wyrazić przypuszczenie, że silna tendencja do standaryzacji, odzwierciedlająca uzasadnione i racjonalne dążenia do oszczędności wysiłku intelektualnego winna doporwadzić do powstania dostatecznie ogólnego języka programowania, pozbawionego rozwiązań prowizorycznych i partykularnych, mającego natomiast cechy umożliwiające rozwój języka bez konieczności zmiany jego zasadniczej struktury. Innymi słowy, wydaje się, że względnie stały standard może być osiągnięty pod warunkiem połączenia języka programowania z jego metajęzykiem. Przy takim rozwiązaniu nie tylko program docelowy, ale i odpowiedni translator byłby generowany w czasie translacji programu źródłowego, składającego się z dwu części (niekoniecznie rozdzielnych leksykograficznie): z definicji języka i z programu w nim napisanego. Nie znaczy to, naturalnie, że każdy program musi zawierać *explicite* pełną definicję języka, można przyjąć, że pewien lokalny (np. dla danego ośrodka obliczeniowego) standard języka zostanie ustalony i indywidualne programy zawierać będą tylko informację opisującą odchylenia od tego standardu. Istotną zaletą takiego systemu byłoby, że każdy „lokalny standard" może być opisany w ramach standardu uniwersalnego. Dodajmy, że pewne próby utworzenia takiego systemu zostały podjęte w ostatnich latach (por. Uogólniony Algol A. van Wijngaardena [108] oraz sekcję 3.3.3 niniejszej książki).

Podział języków programowania

Istniejące języki programowania dzielimy w zasadzie ze względu na rodzaj zastosowań jakim służą.

1. *Języki algorytmiczne* (*procedure-oriented languages*). Do grupy tej należą języki o roz-

budowanej składni umożliwiającej opis różnorodnych procesów przetwarzania informacji, przy czym ani przedmiot przetwarzania, ani przyjęty algorytm nie są w żaden sposób ograniczone ani narzucone przez formalizm języka. Jednakże, dla pewnych zastosowań maszyn matematycznych wygodniejsze są mniej ogólne języki programowania, mające za to szereg specjalnych ułatwień dla konkretnego rodzaju zastosowań i zespołu użytkowników (patrz poniżej). Języki algorytmiczne dzielimy dalej ze względu na typ zastosowań, dla których dany język jest szczególnie przydatny:

a. *Języki algebraiczne* (np. Algol-60, Fortran) są to języki przeznaczone głównie do zapisywania algorytmów numerycznego przetwarzania informacji.

b. *Języki ekonomiczne* (np. COBOL) są to języki przeznaczone głównie do opisywania algorytmów przetwarzania informacji o charakterze handlowym, bankowym i statystycznym. W odróżnieniu od języków algebraicznych mają bardzo rozbudowane mechanizmy operacji wejścia/wyjścia oraz pewne możliwości wykonywania operacji na danych nienumerycznych.

c. *Języki do symbolicznego przetwarzania informacji* (np. LISP, IPL V, por. [83]) są to języki przeznaczone głównie do zapisywania algorytmów przetwarzania informacji nienumerycznych, mają szeroko rozbudowane mechanizmy wykonywania operacji na danych symbolicznych.

2. *Języki symulowania* (np. SOL) są to języki ułatwiające opisywanie algorytmów modelowania i symulacji, mają rozbudowany mechanizm automatycznej sprawozdawczości, pewien stopień równoległości działania i środki pozwalające analizować zależności czasowe takich działań.

3. *Języki bezpośredniego dostępu* – są to języki pozwalające na wprowadzenie zasady „konwersacji" z maszyną cyfrową (por. sekcja 4.1.4).

4. *Języki problemowe* (*problem-oriented languages*) – są to języki ściśle dostosowane do wyodrębnionej klasy zadań; mają z reguły ubogą składnię, formalizm zapisu implikuje metodę postępowania. Program napisany w takim języku składa się z szeregu werbalnych instrukcji (słów kluczowych) uzupełnionych dodatkowymi informacjami, noszącymi charakter parametrów konkretnego problemu. Autorzy języków problemowych wkładają wiele wysiłku w to by jak najpełniej dostosować zestaw słów kluczowych do żargonu zawodowego grupy użytkowników, dla której język jest przeznaczony, oraz by zminimalizować ilość informacji dostarczanej przez użytkownika przy rozwiązywaniu konkretnego zadania. Siłą rzeczy języki tego typu nie nadają się do formułowania żadnych istotnie nowych algorytmów; programowanie sprowadza się do kombinowania ustalonych procedur w sposób niezbędny do rozwiązania określonego problemu.

Języki problemowe przyjmują niekiedy trochę inną postać od naszkicowanej powyżej. Jedną z niej są tzw. *pakiety programowe*.

Pakiety programowe stanowią bibliotekę programów opracowanych najczęściej przez producenta maszyny na specjalne zamówienie nabywcy, poświęconą określonemu zespołowi zadań przetwarzania informacji. Biblioteka taka, wraz z odpowiednim programem sterującym stanowi kompletne wyposażenie programowe użytkownika. Różnica między językiem problemowym opisanym uprzednio i systemem pakietów problemowych jest niezbyt istotna, dotyczy głównie najbardziej zewnętrznej formy wykorzystania systemu: przy syste-

SPIS CYTOWANEJ LITERATURY

Oprócz ogólnie stosowanych skrótów przyjęto następujące oznaczenia:

CA = Computers and Automation
CACM = Communications of the ACM
JACM = Journal of the ACM
Comp. J. = The Computer Journal

[1] Arbuckle, R. A., *Computer Analysis and Thruput Evaluation*, CA, 15, No 1 (1966), str. 12–15, 19.

[2] Arden, B. W., Galler, B. A., O'Brien, T. C., Westervelt, C. H., *Program and Addressing Structure in a Time-Sharing Environment*, JACM, 13 (1966), str. 1–16.

[3] Backus, J. W., *The Syntax and Semantics of the Proposed International Algebraic Language of the Zürich ACM–GAMM Conference May 1958*, Proceedings of the IFIP Congress 59, Paris (1960), str. 125–132.

[4] Ball, N. A., Foster, H. Q., Long, W. H., Sutherland, I. E., Wigington, R. J., *A Shared Memory Computer Display System*, IEE Trans., EC-15 (1966), str. 150–156.

[5] Bernstein, A. J., *Analysis of Programs for Parallel Processing*, IEE Trans., EC-15 (1966), str. 751–763.

[6] Brooker, R. A., *The Autocode Programs Developed for the Manchester University Computers*, Comp. J. 1 (1958), str. 15–21.

[7] Brooker, R., Morris, D., *An Assembly Program for a Phrase Structure Language*, Comp. J., 3 (1960), str. 168.

[8] Brown, W. S., *An Operating Environment for Dynamic-Recursive Programming Systems*, CACM, 8 (1965), str. 371–376.

[9] Buchholz, W. (Editor), *Planning a Computer System*, New York 1962.

[10] Carraciolo di Forino, A., *Linguistic Problems in Programming Theory, Proceedings of the IFIP Congress 65*, Washington 1965, str. 223–228.

[11] Chapin, N., *Logical Design to Improve Software Debugging – A Proposal*, CA, 15, No 1 (1966), str. 22–24.

[12] Codd, L. F., *Multiprogramming, Advances in Computers*, 3 (1962), str. 77–153.

[13] Colilla, R. A., *Time Sharing and Multiprocessing Terminology*, Datamation, 12, No 3 (1966), str. 49–51.

[14] Collin, A. J. T., *A Simple Program for Use in the „conversational mode"*, Comp. J. 9 (1966), str. 238–241.

[15] Corbato, F. J., Dagett, M. M., Daley, R. C., Creasy, R. J. Hellwig, J. D., Orenstein, R. H., Korn, L. K., *The Compatible Time-Sharing System*, Cambridge, Mass., 1963.

[16] Critchlow, A. J., *Multiprogramming and Multiprocessing*, IEEE Spectrum, March (1964), str. 192-198.

[17] Curtin, W. A., *Multiple Computer Systems, Advances in Computers*, 4 (1963) str. 245–303.

[18] Dijkstra, E. W., *Making a Translator for ALGOL*-60, A. P. I. C. Bull. 7 (1961), str. 3–11.

[19] Dijkstra, E. W., *Programming Considered as a Human Activity*, Proceedings of the IFIP Congress 65, Vol. 1, str. 213–217.

[20] Dijkstra, E. W., *Solution of a Problem in Concurrent Programming Control*, CACM, 8 (1965), str. 569.

[21] Dines, R. S., *Telecommunications and Supervisory Control Programs*, CA, 15, No 5 (1966),str. 22—23.

[22] Dunn, T. M., Morrisey, I. H., *Remote Computing — an Experimental System*, Part 1: *External Specification, Proceedinges of the Spring Joint Computer Conference*, (1964), str. 413—423.

[23] Empacher, J., Greniewski, M., Solich, R., Turski, W., Zdanowska, J., *Instrukcja programowania w języku KLIPA*, Warszawa 1964.

[24] Feldman, A., *A Formal Semantics for Computer Languages and its Application in a Compiler-Compiler*, CACM, 9 (1966), str. 3—9.

[25] Fernbach, S., *Computer in the USA — Today and Tomorrow*, Proceedings of the IFIP Congress 65, str. 77—85.

[26] Fischer, F. P., Swindle, G. F., *Computer Programming Systems*, New York 1964.

[27] Flores, I., *Multiplicity in Computer Systems*, CA 15, No 7 (1966), str. 19—23.

[28] Flowers, B. H., *A Report of the Joint Working Group on Computers for Research*, London 1966.

[29] Floyd, R. W., *An Algorithm for Coding Efficient Arithmetic Operations*, CACM, 4 (1961), str. 42—51.

[30] Floyd, R., *Bounded Context Syntactic Analysis*, CACM, 7 (1964), str. 62—67.

[31] Frielink, A. B. (Editor), *Economics of Automatic Data Processing*, Amsterdam 1965.

[32] Garwick, J. V., *GARGOYLE, a Language for Compiler Writing*, CACM, 7, (1964), str. 16—20.

[33] Garwick, J. V., *Gargoyle User's Manual*, Forsvarets Forskningsinstitutt Intern Rapport S-24, 1965.

[34] Gee, D. W. M., *A Data Communications Controller Some Desirable Features*, CA, 15, No 5 (1966), str. 18—20, 50.

[35] Geldard, F. A. (Editor), *Communication Processes*, London 1965.

[36] Gill, S., *Parallel Programming*, Comp. J. 1 (1956), str. 2—10.

[37] Gill, S., *Current Theory and Practice of Automatic Programming*, Comp. J. 2 (1959), str. 110—114.

[38] Gilmore, P. C., *An Abstract Computer with a Lisp-Like Machine Language without a Label Operator* — artykuł w: Bradford, P., Hirschberg, D. (Editors), *Computer Programming and Formal Systems*, Amsterdam 1963.

[39] Greif, H. D., *The Interaction of Hardware Software and Future Developments at TRW Systems*, CA, 15, No 2 (1966), str. 14—18.

[40] Greniewski, M., *Wstęp do programowania i modelowania cyfrowego*, Warszawa 1961.

[41] Gruenberger, F., *Are Small Free-Standing Computers here to Stay?*, Datamation, 12, No 4 (1966), str. 67—68.

[42] Halstead, M. H., *Machine Independent Computer Programming*, Washington 1962.

[43] Hellerman, H., *Parallel Processing of Algebraic Expressions*, IEE Trans, EC-15, 1 (1966), str. 82—91.

[44] Holstein, D., *Computer-Aided Design*, Product Engineering, 23 November, 1964.

[45] Huskey, H. D., *Compiling Techniques for Algebraic Expressions*, Comp. J. 4 (1961), str. 10—19.

[46] Huskey, H. D., *An Introduction to Procedure-Oriented Languages*, Advances in Computers, 5 (1964), str. 349—377.

[47] Hutchinson, G. H., *A Computer Center Simulation Project*, CACM, 8 (1965), str. 559—568.

[48] *IBM 7090 Principles of Operation*, IBM Systems Reference Library A 22—6528—5.

[49] *IBM Operating System/360, PL/I Language Specifications*, IBM Form C 28—6571—0 (1965).

[50] Ingerman, P. Z., *A Syntax-Oriented Translator* New York 1966.

[51] Irons, E., *A Syntax Directed Compiler for ALGOI 60*, CACM, 4 (1961), str. 51—55.

[52] Irons, E. T., *The Structure and Use of the Syntax Directed Compiler*, Annual Review of Automatic Programming 3 (1963), str. 207—227.

[53] Jerszow, A. P. (Ершов, А. П.), *Программирование арифметических операторов*, ДАН, 118 (1958), str. 427—430.

[54] Jerszow, A. P. (redaktor) (Ершов, А. П.), *Автоматизация программирования*, Москва 1961.

[55] Jerszow (Ershov), A. P., Rar, A. F., *SYGMA — A Symbolic Generator and Macroassembler*, referat wygłoszony na Sympozium IFIP w Pizie, wrzesień 1966.

[56] Jewreinow, E. W. (Евреинов Э. В.), *Универсальные вычислительные системы с частично переменной структурой*, Вычислительные системы, 17 (1965), str. 3—60.

[57] Jewreinow, E. W., Kosariew, J. G. (Евреинов Э. В., Косарев Ю. Г.), *Матричный р-язык дла описания параллельных автоматов*, Вычислительные системы, 17 (1965), str. 100—105.

[58] Jewreinow, E. W., Kosariew, J. G. (Евреинов Э. В., Косарев Ю. Г.), *О решении задач на универсалиых системах*, Вычислительные системы, 17 (1965), str. 106—164.

[59] Kagan, B. M., Ter-Mikaelian, T. M. (Каган, Б. М., Тэр-Микаэлян Т. М.), *Решение инженерных задач на цифровых вычислителных машинах*, Москва 1964.

[60] Kalenich, W. A. (Editor), *Proceedings of the IFIP Congress 65*, Washington 1965--1966.

[61] Heller, J, M., Strum, E. C., Yang, G. H., *Remote Computing — An Experimental System* Part 2: *Internal Design*, Proceedings Spring Joint Comp. Conf. (1964), str. 425-443.

[62] Kilburn, T., Payne, R. B., Howarth, D. J., *The Atlas Supervisor*, Computers — Key to Total Systems Control, New York 1962.

[63] Kinslow, H. A., *The Time-Sharing Monitor System*, Proceedings Spring Joint Comp. Conf. (1964), str. 443-454.

[64] Kitow, A., Krinicki, N., *Elektroniczne maszyny cyfrowe oraz programowanie*, Warszawa 1963.

[65] Krinicki, N. A., Mironow, G. A., Frołow, G. D. (Криницкий, Н. А, Миронов, Г. А. Фролов, Г. Д.), *Программирование*, Москва 1966.

[66] Knight, K. E., *A Fast Sort of Country*, rozprawa doktorska cytowana w [44.]

[67] Knuth, D. E., McNaley, J. L., a) *SOL — A Symbolic Language for General Purpose Simulation*, b) *A Formal Definition of SOL*, IEE Trans. EC 13 (1964), str. 401-414.

[68] Kosariew, J. R. (Косарев, Ю. Р.), *О методике решения задач на универсальных вычислителных системах*, Вычислительные системы, 17 (1965), str. 61-99.

[69] Krishnamoorthi, B., Wood, R. C., *Time-Shared Computer Operations with Both Interarrival and Service Times Exponential*, JACM, 13, No 3 (1966), str. 317-338.

[70] Kurtz, T. E., *New Demands on Software*, CA, 15, No 2 (1966), str. 22-13, 54.

[71] Ledley, R. S., *Programming and Utilizing Digital Computers*, New York 1962.

[72] Ledley, R. S., *Optical Processing in Medical Sciences*, CA, 15, No 7 (1966), str. 14—18, 23.

[73] Lehmer, D. H., *Some High Speed Logic*, Proceedings of the Symposium on Applied Mathematics, American Math. Society, 15, 1963.

[74] Licklider, J. C. R., *Problems in Man-Computer Communication*, artykuł w: Geldard, F. A., (Editor), *Communications Processes Proceedings of a Symposium held in Washington* (1963), str. 221-268.

[75] Ławrow, S. S. (Лавров, С. С.), *Об экономии памяти в замкнутых операторных схемах*, Журнал Выч. Мат. и Мат. физики, 1 (1961), str. 678-701.

[76] Mac Gowan, R. A., *Small Scientific Computers versus Data Communications Systems in a Large Computer Environment*, CA, 13, No 2 (1964), str. 24-28.

[77] Mc Carthy, J., *LISP 1.5 Programmer's Manual*, *The* M. I. T. Press, 1962.

[78] Macri, J. F., *Communication-Oriented Computer System*, CA, 15, No 5 (1966), str. 14-16, 50.

[79] Marczuk, G. I., Jerszow, A. P., *Man-Machine Interaction in Solving a Certain Class of Differential Equations*, Proceedings of the IFIP Congress 65, str. 550-551.

[80] *Monsanto Streamlines Computer Methods*, Chemical and Engineering News, 37, March 22, 1965, str. 40-41.

[81] Morrisey, J, H., *The Quicktran System*, Datamation, 11, No 2 (1965),

[82] Naur, P., *The Design of the GIER-Algol Compiler*, Nordisk Tidskrift for Informations--Behandlung, 3 (1963), cz. I, str. 124-140, cz. II, str. 145-166.

[83] Newell, A., *Information Processing Language V Manual*, Englewood Cliffs, N. J. 1961.

[84] Organick, E. I. *A Fortran Primer*, London 1963.

[85] Parnas, D. L., *A Language for Describing the Functions of Synchronous System*, CACM, 9 (1966), str. 72-76.

[86] Paszkowski, S., *Język Algol-60*, Warszawa 1968.

[87] Penny, J. P., *An Analysis, Both Theoretical and by Simulation of a Time-Shared Computer System*, Comp. J. 9 (1966), str. 53-59.

[88] Raphael, B., *The Structure of Programming Languages*, CACM, 9, No 2 (1966), str. 67--71.

[89] Randell, B., Russell, L. I., *ALGOL-60 Implementation*, London 1964.

[90] Samelson K., Bauer, F. L., *Sequential Formula Translation*, CACM, 3 (1959), str. 76--83.

[91] Scherr, A. L., *Time Shraing Measurement*, Datamation, 12, No 4 (1966), str. 22-26.

[92] Schwartz, J., *Large Parallel Computers*, JACM, 13 (1966), str. 25-32.

[93] Seitz, R. N., Reinfelds, J., Clem, P. L., Wood, L. H., *The Amtran System*, Datamation, 12, No 10 (1966), str. 22-27.

[94] *Small Computer Handbook*, Digital Equipment Corporation 1966.

[95] Solomon, M. B., Jr., *Economies of Scale and the IBM System/360*, CACM, 9, (1966), str. 435-440.

[96] Strachey, C., *An Impossible Programme*, Comp. J. 7 (1965), str. 313.

[97] Strachey, C., *A General Purpose Macrogenerator*, Comp. J. 8 (1965), str. 225-234.

[98] Strong, J., Wegstein, J., Fritee, A., Olsztyn, J., Mock, O., Steel, T., *The Problem of Programming Communication with Changing Machines*, CACM, 1 (1958), No 8, str. 12-18, No 9, str. 9-15.

[99] Svejgaard, B., Lindblad, P., *IMP III*, Kobenhavs Universitets Matematische Institut Afdelingen for Informations behandlung, 10.

[100] Trachtenbrot, B. A., *Algorytmy i automatyczne rozwiązywanie zadań*, Warszawa 1961.

[101] Trifonow, N. P., Szura-Bura, M. R. (Трифонов, Н. П. Шура-Бура), M. P. *Системa автоматизации программирования*, Москва 1961.

[102] Turski, W., *Uwagi o budowie i wykorzystaniu języków zewnętrznych*, Warszawa 1964.

[103] Turski, W., *Some Results of Research on Automatic Programming in Eastern Europe*, Advances in Computers, 5 (1964), str. 23-109.

[104] Vickers, T., *The organization of a Computing Centre*, artykuł w: Hall, J. A. (Editor), *Computers in Education*, London 1962.

[105] Wagner, F. V., Granholm, J., *Design of a General-Purpose Scientific Computing Facility*, Procedings of the IFIP Congress 65, vol. 1, str. 283-289.

[106] Warmus, M., *GIER-ALGOL*, Warszawa 1966.

[107] Weizenbaum, J., *ELIZA — A Computer Program for the Study of Natural Language Communication Between Man and Machine*, CACM, 9 (1966), str. 36-43.

[108] Wijngaarden van, A., *Generalized ALGOL, Symbolic Languages in Data Processing*, Proceedings of the Symposium on Formal Languages, Rome 1962, str. 409-419.

[109] Wijngaarden van, A., *Numerical Analysis as an Independent Science*, Nordisk Tidskrift for Informations-Behandlung, 6 (1966), str. 66-81.

SPIS RZECZY