informatyka+

Algorytmika i programowanie

Bazy danych

Multimedia, grafika i technologie internetowe

Sieci komputerowe

Tendencje w rozwoju informatyki i jej zastosowań

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

informatyka+

Wszechnica Popołudniowa:

Tendencje w rozwoju informatyki i jej zastosowań

Język językowi nie równy

Jan Madey

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Język językowi nie równy

Rodzaj zajęć: Wszechnica Popołudniowa
Tytuł: Język językowi nie równy
Autor: prof. dr hab. Jan Madey

Redaktor merytoryczny: prof. dr hab. Maciej M Sysło

Zeszyt dydaktyczny opracowany w ramach projektu edukacyjnego **Informatyka+** — ponadregionalny program rozwijania kompetencji uczniów szkół ponadgimnazjalnych w zakresie technologii informacyjno-komunikacyjnych (ICT).
www.informatykaplus.edu.pl
kontakt@informatykaplus.edu.pl

Wydawca: Warszawska Wyższa Szkoła Informatyki
ul. Lewartowskiego 17, 00-169 Warszawa
www.wwsi.edu.pl
rektorat@wwsi.edu.pl

Projekt graficzny: FRYCZ I WICHA

Warszawa 2011

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

WARSZAWSKA WYŻSZA SZKOŁA INFORMATYKI

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Język językowi nie równy

Jan Madey
Uniwersytet Warszawski, Instytut Informatyki
madey@mimuw.edu.pl

Wstęp

Człowiek od zarania dziejów miał potrzebę komunikowania się z innymi ludźmi, zarówno bezpośrednio, jak i zdalnie. Podobne potrzeby wykazują zresztą i zwierzęta. Różne były formy tej komunikacji, ale niewątpliwie najważniejszą z nich jest *język*, który w swojej wersji pisanej stał się podstawą rozwoju cywilizacji. Językowi są więc poświęcone zaawansowane badania w różnych dyscyplinach naukowych (z *językoznawstwem*, zwanym także *lingwistyką*, na czele). Ale język, będąc narzędziem pracy wielu specjalistów różnych dziedzin, stał się też przedmiotem badań niektórych z nich. Szczególną pozycję ma tutaj *informatyka*, gdyż *języki programowania* odgrywają w tej dyscyplinie kluczową rolę.

W rozdziale pierwszym niniejszego opracowania zajmiemy się wybranymi ogólnymi aspektami lingwistycznymi, a następnie (rozdz. 2 i 3) prześledzimy – w sposób uproszczony oraz subiektywny – ewolucję języków programowania[1].

Spis treści

[1] Rozdziały 2 i 3 niniejszego tekstu są w dużej części zaczerpnięte z pracy [JM04].

1 JĘZYK, MOWA, PISMO, ALFABET

Bez problemu można sprawdzić w Internecie znaczenie tytułowego terminu (przede wszystkim w Wikipedii – wolnej encyklopedii: http://pl.wikipedia.org/wiki/ oraz w WIEM: http://wiem.onet.pl/).

Spójrzmy zatem do mniej obecnie popularnych źródeł papierowych, a w szczególności do tych mało dostępnych (jak na przykład [WEPI]), by sprawdzić, czy i ponad 100 lat temu tak samo lub co najmniej podobnie rozumiało się te pojęcia.

1.1 DEFINICJE SPRZED PONAD WIEKU

Wielka Encyklopedia Powszechna Ilustrowana [WEPI], wydawana w latach 1880-1914 (55 tomów) nigdy nie została ukończona. – została przerwana na haśle „Patroklos", ze względu na I Wojnę Światową. Jej wydawcy, to F. J. Granowski i S. J. Sikorski, ale popularnie jest nazywana „encyklopedią Sikorskiego". Oto jak omówiono w niej interesujący nas termin i inne z nim związane (przedstawiamy skróty, z zachowaniem oryginalnej pisowni):

„**Język i języki**, w znaczeniu lingwistycznym, językoznawczym, w znaczeniu *mowy ludzkiej*. Nazwa ta stosuje się pod wpływem przenośni, przyczym główny organ widomy wykonawczy, główne narzędzie wymawiania, *język*, w znaczeniu całej czynności, w znaczeniu procesu i całości mówienia. [...] Oczywiście nazwa ta zjawiła się wtedy, kiedy język, w ustach ludzkich umieszczony, stał się istotnie głównym narzędziem wymawiania. W pierwocinach ludzkości, kiedy wymawianie było umiejscowione głębiej, przeważnie w krtani, trudniej mogłoby się dokonać podobne rozszerzenie nazwy języka. – Skojarzenie nazwy języka, jako głównego narzędzia wymawiania, z nazwą języka, jako mowy ludzkiej, widoczne jest dotychczas w takich wyrażeniach, jak np. polskie: „długi język", „miele językiem", „na końcu języka", „ciągnąć kogo za język", „język mu skołowaciał", „dostać się na języki" [...].– W dalszym ciągu „*język*" znaczy: wiadomość, doniesienie, [...] itd.; np. „zasięgnąć języka", „z językiem przybieżał" [...].– Następnie J. jest to mowa, która odróżnia, z jednej strony, człowieka od człowieka (mówienie indywidualne), plemię od plemienia, naród od narodu (język w ścisłym znaczeniu), z drugiej zaś strony, ludzkość czyli rodzaj ludzki w ogóle (mowa ludzka) od wszystkich innych zwierząt. [...] W znaczeniu języka plemiennego lub narodowego synonimami *języka* lub też wyrazami blizko znacznemi są: *mowa, żywe słowo, narzecze, gwara, dyjalekt* itp., a w znaczeniu języka indywidualnego – *styl*, *sposób mówienia*, itp. („język ostry, zuchwały, rozwiązły, pospolity, banalny, kwiecisty, wzniosły [...]"). Przenośnie mówimy: „J. kwiatów", „J. przyrody", w znaczeniu zaś bardziej zbliżonym do języka ludzkiego – „J. zwierząt". – J., w najobszerniejszym znaczeniu tego wyrazu, w znaczeniu przenośnym, oznacza wszelkie sposoby porozumiewania się, a więc wszelką mimikę i giesty (język palców itp.), pismo (język znaków itp.), głosy mimowolne i odruchowe [...]. – Obok tego zwykłego pojmowania społeczno-towarzyskiego można J. określać z rozmaitych stanowisk naukowych. Dla fizjologa mówienie i język wogóle jest funkcją organizmu ludzkiego [...]. Dla antropologa, jako przyrodnika, język jest jedną z cech, wyróżniających rodzaj ludzki z pomiędzy całego szeregu istot podobnie organizowanych. Nareszcie dla psychologa język jest usystematyzowanym zbiorem wyobrażeń, a więc zjawiskiem o podstawie wyłącznie psychicznej [...]." [WEPI] XXXIII, 1903, str. 266-278].

Mowa – niestety nie udało nam się dotrzeć do tego hasła w [WEPI]. Ale w wyżej cytowanych zapisach pojawił się ten termin kilkakrotnie i jest traktowany jako synonim języka mówionego: „– Skojarzenie nazwy języka, jako głównego narzędzia wymawiania, z nazwą języka, jako **mowy** ludzkiej, widoczne jest [...]", „Następnie J. jest to **mowa**, która odróżnia [...]", „W znaczeniu języka plemiennego lub narodowego synonimami *języka* lub też wyrazami blizko znacznemi są: ***mowa***, [...]"

Pismo – jak już wiemy, encyklopedia Sikorskiego została przerwana na haśle „Patrokles". Zauważmy jednak, podobnie jak powyżej, że w określeniu pojęcia *język*, pojawia się też termin *pismo*: „– J., w najobszerniejszym znaczeniu tego wyrazu, w znaczeniu przenośnym, oznacza wszelkie sposoby porozumiewania się, a więc wszelką mimikę i giesty (język palców itp.), **pismo** (język znaków itp.), głosy mimowolne i odruchowe [...]."

Pozostał już tylko alfabet. Takie zapisy o alfabecie znajdują się w naszej encyklopedii (por. [WEPI] I, 1890, str. 668 i str. 32-36):
„**Alfabet**, ob. Abecadło".

„**Abecadłem** nazywa się zbiór wszystkich znaków piśmiennych, albo liter na wyrażenie pojedyńczych dźwięków pewnego języka, ułożony w pewien zwyczajem ustalony porządek. Nazwa ta wzięła początek od brzmienia w mowie naszej trzech pierwszych głosek *a b c* w połączeniu z przyrostkiem *–dło* (pochodnym *–adło*). Dawniejszą od tego wyrazu jest nazwa grecka *alfabet*, utworzona z nazw dwóch początkowych liter w abecadle greckim *alfa*, *beta*, odpowiadającym naszym *a b*. [...]"

1.2 NIECO WSPÓŁCZESNOŚCI

Z zacytowanych wyżej tekstów wynika, że nie uległo jakiejś istotnej zmianie spojrzenie na interesujące nas terminy. Język, jako narząd w jamie ustnej, biorący przede wszystkim udział w ssaniu, żuciu, połykaniu pokarmów, poprzez swoją dodatkową funkcję istotną w procesie komunikowania się ludzi stał się w sposób naturalny wyrazem używanym dla określania różnych form tej komunikacji. Stąd i następujące znaczenia tego terminu ([SJP78], tom pierwszy, str. 844):

- «zasób wyrazów, zwrotów i form określanych przez reguły gramatyczne, funkcjonujący jako narzędzie porozumiewania się przez członków jednego narodu, społeczeństwa; mowa»: Język polski, angielski, francuski. Język ojczysty. Języki obce [...]. ΔJęzyki starożytne, klasyczne [...] ΔJęzyk literacki [...].
- «zasób wyrazów, zwrotów i form, używanych w celu porozumienia się przez ludzi pewnego środowiska, zawodu, regionu; gwara, dialekt, żargon»: Język techniczny, dyplomatyczny, łowiecki, uczniowski, złodziejski. [...]
- «sposób wyrażania się, wysławiania się, charakterystyczny dla danego autora, dzieła, epoki; styl»: Język Prusa. Język utworu [...] ΔJęzyk książkowy, pisany, ΔJęzyk potoczny, mówiony
- *dawn*. «jeniec schwytany dla powzięcia wiadomości o nieprzyjacielu»

Możemy jeszcze wskazać wiele innych zwrotów, w których pojawia się wyraz *język* w zbliżonym znaczeniu: język filmu, tłumu, dźwięków, nauki, migowy, biznesu, uczuć itd. Ważny jest także podział na *języki naturalne* i *języki sztuczne*. Tym ostatnim poświęcimy więcej uwagi w dalszych rozdziałach.

Zauważmy teraz, że w najszerszym lingwistycznie znaczeniu termin *język* oznacza zarówno *język mówiony*, czyli *mowę*, jak i *język pisany*, czyli *pismo*. Z kolei mówiąc o piśmie w naturalny sposób dochodzimy do terminu *alfabet*. Trzeba przy tym zauważyć, że rola języka pisanego jest nie do przecenienia, co oddaje m.in. poniższy tekst ([DD72], str. 24]):

„Pismo jest graficznym odbiciem języka. Każdy element systemu pisma: znak pisarski, litera, 'hieroglif', słowo pisane, ma swój odpowiednik w określonym elemencie systemu pierwotnego, czyli mowy [...]. Pismo jest więc naturalną metodą, czy raczej najważniejszą z naturalnych metod przekazywania mowy i umożliwia wymianę myśli ludziom, którzy nie mogą się posłużyć mową na skutek odległości w czasie lub przestrzeni bądź z jakiś innych przyczyn."

Bez pisma nie byłby możliwy cywilizacyjny rozwój człowieka. Ale droga do pojawienia się pisma współczesnego była długa i ciernista, a konsekwencje niektórych historycznie błędnych decyzji są ponoszone do dzisiaj, co obrazuje poniższa krótka historia pisma japońskiego.

1.3 PRZYKŁAD HISTORYCZNEJ BŁĘDNEJ DECYZJI – PISMO JAPOŃSKIE (KANJI)

Mówiony język japoński ma klarowną strukturę (choć bardzo odmienną od tej, do której jesteśmy przyzwyczajeni) oraz łatwą – dla nas, Polaków – wymowę. Inaczej jednak wygląda sprawa pisma. Zamiast opracować własne, Japończycy – będąc niejako pod względem kulturowym „kolonią" Chin – od około III/IV wieku n.e., zaczęli przejmować chińskie pismo, które zupełnie nie pasowało do ich języka mówionego. Proces ten trwał kilkaset lat i w efekcie powstał bardzo skomplikowany system, obejmujący zarówno ideogramy chińskie (*kanji*), jak i dwie postaci pisma sylabicznego: *hiragana* i *katakana*. Ideogramy wykorzystuje się tylko do zapisywania rzeczowników, przymiotników i pni czasownikowych. Ale japoński ma również końcówki gramatyczne (których brakuje w języku chińskim), przyimki itd., i do ich wyrażania używa się *hiragany*. Natomiast *katakana* służy przede wszystkim do transliterowania nazw własnych (np. nazwisk europejskich) oraz słów obcego pochodzenia.

To jeszcze nie koniec. Każdy znak *kanji* ma w języku japońskim dwie odrębne wymowy: *kun* (japońską) i *ON* (chińską, zwaną też sino-japońską), przy czym ta ostatnia ma niewiele wspólnego ze współczesnym mówio-

nym językiem chińskim – brzmi tak, jak Japończycy słyszeli wtedy, gdy znaki te były przyswajane. Nie ma przy tym jednoznacznych zasad określających kiedy stosuje się którą wymowę. Zgrubnym kryterium jest to, czy dany znak jest samodzielną jednostką znaczeniową (wtedy czyta się go na ogół „po japońsku"), czy też dopiero połączony z innymi taką tworzy (wówczas czyta się go zwykle „po chińsku"). Ponieważ wymowa chińska była i jest różna w różnych prowincjach i ponadto zmieniała się w rozmaitych okresach historycznych, to znak *kanji* ma zazwyczaj kilka wersji czytania *ON*.

Dla ilustracji rozważmy jeden z najprostszych znaków *kanji* – ideogram oznaczający człowieka:

Jego wymowa japońska to *hito*, zaś chińska to *JIN* lub *NIN*.

Weźmy teraz pod uwagę ideogram oznaczający duży (człowiek z rozpostartymi rękoma):

Tutaj mamy następujące wymowy: *ookii* (japońska) oraz *TAI*, *DAI* (chińska).

Zastanówmy się teraz, co oznacza następujące połączenie obu tych znaków w jedną jednostkę znaczeniową:

Jest to duży człowiek, czyli *dorosły*. No i teraz najważniejsze pytanie – jak tę kombinację znaków odczytujemy? Gdyby zastosować wspomnianą wcześniej regułę, to powinno to być *taijin* lub *tajnin*, ew. *daijin* bądź *dainin*. A może jednak *taihito* lub *daihito*? Nic z tych rzeczy! Te dwa ideogramy zapisane razem wymawia się *otona*, czyli bez żadnego związku z japońską i chińską wymową znaków składowych. Dlaczego? Ponieważ po japońsku dorosły to właśnie *otona*...

Powyższe fakty ilustrują tezę, iż przyjęte w Japonii pismo nie pasuje do mowy japońskiej. Dodajmy do tego, że liczebność *kana*[2] (czyli *hiragany* i *katakany*), to 100 znaków, ale już jeśli chodzi o *kanji*, to mamy do czynienia z tysiącami ideogramów (absolwent szkoły musi znać ich około 2000), z których – jak już wiemy – każdy ma czytanie japońskie i chińskie, niekiedy w kilku wariantach. Do tego dochodzi kłopot z samą pisownią – znaki są charakteryzowane „dynamicznie", tzn. trzeba znać kolejność pisania każdej kreski wchodzącej w skład ideogramu oraz jej kierunek (lewo-prawo czy odwrotnie, góra-dół czy odwrotnie itd.). A liczba kresek w znaku może przekroczyć nawet 20.

[2] Znaki *kana*, to nie jest prawdziwe pismo sylabiczne, ponieważ nie używa się ich samodzielnie – choć były takie próby – lecz tylko wskazuje się za ich pomocą cechy fleksyjne, takie jak czasy czasowników, przyimki itp., podczas gdy same rzeczowniki, czasowniki i przymiotniki wyraża się zawsze nadal znakami *kanji*.

Aż podziw bierze, że Japończycy są w stanie nauczyć się w szkole czegokolwiek więcej poza własnym pismem!

1.4 JĘZYKI NATURALNE A JĘZYKI SZTUCZNE

Uważa się, że na świecie istnieje obecnie około 2500-3000 języków, a po wielu innych nie pozostało już śladu. Próbą „ludowej" odpowiedzi na pytanie o pochodzenie wielości języków jest opowieść o wieży Babel: „**Wieża Babel**, według biblijnej *Księgi Rodzaju* wieża, jaką zaczęto wznosić w ziemi Szinear w Babilonie (po hebrajsku Babel) z zamiarem, by sięgnęła nieba. Rozgniewany zuchwałością budowniczych Bóg sprawił, że zaczęli mówić różnymi językami i nie mogąc się porozumieć, musieli przerwać budowę." (por. [WIEM]).

Języki naturalne rodziły się niejako spontanicznie, ale były też różne próby opracowania języka uniwersalnego, który stałby się międzynarodowym standardem. Kończyły się one na ogół fiaskiem, a rolę takiego standardu w różnych okresach przejmowały do pewnego stopnia modne bądź użyteczne konkretne języki naturalne – obecnie jest to niewątpliwie angielski. Z języków sztucznych niewątpliwie na uwagę zasługuje *esperanto*. Podstawy tego języka, zwanego pierwotnie *Lingvo Internacia* (pol. *język międzynarodowy*), przedstawił Ludwik Zamenhof[3] pod pseudonimem Dr. Esperanto, w 1887 roku. Główną ideą Zamenhofa było opracowanie neutralnego oraz łatwego do nauki języka, przydatnego do międzynarodowej komunikacji, nie zastępującego jednak narodowych języków. Choć cel ten nie został nigdy w pełni osiągnięty, to esperanto jest używane przez międzynarodową wspólnotę (której wielkość jest szacowana nawet na dwa miliony osób), podczas kongresów i dyskusji naukowych, a także w muzyce, teatrze, kinie i mediach prasowcyh. Esperanto doczekało się także formalnego międzynarodowego uznania w postaci dwóch rezolucji UNESCO.

W dalszej części niniejszego opracowania skoncentrujemy się na ważnej grupie języków sztucznych, a mianowicie na *językach programowania*. Zauważmy na wstępie, że w wypadku języków naturalnych, mowa poprzedza pismo, czyli w pewnym sensie semantyka jest pierwotna, a syntaktyka wtórna – do wypowiedzi, których znaczenie jest zrozumiałe dobieramy sposób ich zapisu. Inaczej jest w wypadku języków sztucznych. Tam często zaczynamy od zdefiniowania zapisu (składni), a potem objaśniamy mniej lub bardziej ściśle jego semantykę.

2. EWOLUCJA JĘZYKÓW PROGRAMOWANIA

Przez język programowania rozumie się powszechnie *notację* do zapisu *algorytmów* w celu ich wykonania przez *komputer*. Notacja ta powinna być możliwie dogodna dla człowieka, ale jednocześnie precyzyjna i jednoznaczna, ze ściśle określoną *składnią* (*syntaktyką*) oraz *znaczeniem* (*semantyką*). Algorytm zapisany w języku programowania, czyli *program*, musi być następnie przetłumaczony na język wykonywalny przez procesor komputera, tzn. na jego *język wewnętrzny*. Proces tłumaczenia przebiega albo jednorazowo, dla całego programu – mówimy wówczas o *kompilacji*, albo etapami, fraza po frazie – mamy wówczas *interpretację*. Stąd i narzędzia realizujące ów proces (*translatory*) nazywa się odpowiednio *kompilatorami* lub *interpretatorami*. Tekst w języku programowania nosi miano *kodu źródłowego*, zaś jego odpowiednik po translacji, to *kod wynikowy* (lub *kod maszynowy*).

2.1 WCZESNE JĘZYKI PROGRAMOWANIA

Komputery *sensu stricto* pojawiły się w latach czterdziestych ubiegłego wieku i od tego czasu zaczyna się także historia rozwoju języków programowania. Algorytmy były wówczas zapisywane w językach wewnętrznych (zwanych też *językami niskiego poziomu*), tzn. formułowane w kategoriach *rozkazów* – inaczej instrukcji – procesorów konkretnych komputerów. A repertuar takich rozkazów był i nadal jest dosyć ubogi – w pewnym uproszczeniu można powiedzieć, że składa się on z różnych poleceń organizacyjnych (typu: „przenieś zawartość rejestru X do rejestru Y", czy „przesuń bity w lewo") oraz z prostych operacji arytmetycznych i logicznych.

[3] Warto zauważyć, że Ludwik Zamenhof (właściwie Eliezer Lewi Samenhof, ur. 15 grudnia 1859 w Białymstoku, zm. 14 kwietnia 1917 w Warszawie, lekarz), już w wieku 10 lat napisał dramat Wieża Babel, czyli tragedia białostocka w pięciu aktach; uważał bowiem, że główną przyczyną nieporozumień i sporów między ludźmi jest bariera językowa. Jeden, wspólny język miał być jej rozwiązaniem.

Zarówno instrukcje, jak i ich argumenty, są przy tym reprezentowane jako ciągi zerojedynkowe (*binarne*). Innymi słowy alfabet tych języków był bardzo ubogi – składał się z dwóch znaków. Co więcej, ten sam ciąg binarny umieszczony w konkretnym miejscu pamięci komputera, może być w ramach tego samego programu zinterpretowany raz jako argument, a raz jako rozkaz – to był właśnie genialny pomysł Johna von Neumanna, pozwalający w szczególności na formułowanie programów, które mogą się same modyfikować!

Jak wyglądają podane we wstępie nasze trzy atrybuty? *Notacja*, to – jak już wiemy –ciągi zerojedynkowe. Szybko jednak zamieniono ją na nieco bardziej czytelną formę, tzn. dopuszczono używanie znaków alfanumerycznych, przyjmując pewne skróty mnemotechniczne dla oznaczenia rozkazów oraz zapisując liczby w systemie dziesiętnym. Oznaczało to wszakże konieczność tłumaczenia takich tekstów na postać binarną, ale wobec zależności „jeden do jednego" było to zadanie banalne i bez trudu opracowano pierwsze asemblery (pod tym terminem rozumie się zarówno język mnemotechniczny, jak i jego translator).

Komputery tamtych czasów były bardzo dużymi, kosztownymi i zawodnymi urządzeniami. Programowane *algorytmy* – zwłaszcza w latach II Wojny Światowej oraz bezpośrednio powojennych – wiązały się z próbą rozwiązania różnych problemów naukowych o strategicznym znaczeniu przede wszystkim dla celów militarnych. Stąd dostęp do komputerów miała tylko wyselekcjonowana grupa ekspertów i kwestia wygody notacyjnej przy programowaniu miała tym samym drugorzędne znaczenie – inne sprawy były znacznie ważniejsze. Ale taka sytuacja nie trwała długo.

2.2 PIERWSZE JĘZYKI WYSOKIEGO POZIOMU

W połowie ubiegłego wieku zaczęły się pojawiać *języki programowania wysokiego poziomu*. Początkowo chodziło jedynie o wprowadzenie bardziej czytelnej notacji, bez zmiany pojęć używanych do zapisu algorytmu. Później zaczęto nieco uogólniać i modyfikować te pojęcia, ale nadal było to jeszcze *podejście wstępujące* (ang. *bottom-up*) w tym sensie, że w tle były rozwiązania techniczne konkretnego komputera, a programowany algorytm należało wyrażać w kategoriach udostępnianych przez język wewnętrzny tegoż komputera.

Do najbardziej znanych języków tej klasy należy niewątpliwie *Fortran* (***FOR****mula* ***TRAN****slator*), opracowany w latach 1954-57 przez zespół z firmy IBM pod kierunkiem Johna Backusa, w celu ułatwienia korzystania z komputerów serii IBM 704. Architektura tego konkretnego komputera odbiła swoje piętno na rozwiązaniach przyjętych w języku Fortran (dotyczy to np. postaci instrukcji warunkowej, czy też nazw kanałów wejściowych). Język ten przeszedł w następnych latach dużą metamorfozę – powstał m.in. *Fortran IV*, *Fortran 77*, *Fortran 90*, *Fortran 95* oraz *Fortran 2003* (!). Fortran jest więc nadal popularny, zwłaszcza w gronie osób programujących algorytmy dotyczące obliczeń naukowych, a przede wszystkim problemów numerycznych. W trakcie prac nad językiem Fortran Backus wymyślił pierwszą wersję swojej notacji *BNF*[4], którą wykorzystano do opisu składni tego języka. W nieco zmodyfikowanej przez Petera Naura postaci notacja ta została użyta do opisu języka Algol 60, który omawiamy dalej.

Przełom nastąpił pod koniec lat pięćdziesiątych, gdy międzynarodowe grono specjalistów z Europy i Ameryki Północnej rozpoczęło prace nad językiem *Algol 60* (po doświadczeniach z jego nieco ułomnym pierwowzorem, językiem *Algol 58*). Już sama nazwa tego języka, będąca skrótem od słów ***ALGO****rithmic* ***L****anguage*, wskazywała na zmianę podejścia – tym razem jego twórcom chodziło o zdefiniowanie notacji do formułowania algorytmów, nie mającej żadnego związku z architekturą konkretnego komputera. Innymi słowy zmieniono dotychczasowe podejście wstępujące na *zstępujące* (ang. *top-down*) i skoncentrowano się na obiektach oraz operacjach potrzebnych do wyrażania algorytmów, które później byłyby automatycznie wykonywane. Język Algol 60 występował przy tym w tzw. *wersji wzorcowej*, mającej formalnie opisaną składnię (w notacji BNF, co należy uznać za osobisty sukces Petera Naura) i rygorystycznie – choć tylko w języku naturalnym – znaczenie oraz w *konkretnych realizacjach*, tzn. w wersjach powiązanych już z kompilatorami dla konkretnych komputerów. Przykładem bardzo udanej takiej konkretnej realizacji (implementacji) języka Algol 60 był *Gier Algol* w swoich kolejnych edycjach, opracowany przez Naura dla duńskiego komputera GIER.

[4] Akronim ten oryginalnie oznaczał Backus Normal Form. Później jednak, dla podkreślenia zasług Petera Naura w spopularyzowaniu tej notacji przy okazji opisu Algolu, przyjęto traktować BNF jako skrót nazwy Backus–Naur Form.

Niezależnie od prac nad językiem Algol, grupa osób w USA pod kierunkiem Grace Murray Hopper, reprezentująca różne organizacje (w tym konkurujące wytwórnie komputerów), rozpoczęła w 1959 roku prace zmierzające do zaprojektowania języka maszynowo-niezależnego. Tym razem chodziło jednak o przetwarzanie danych, czyli o przekształcanie zbiorów danych alfanumerycznych oraz tworzenie różnych raportów, ale przy tym w sposób zrozumiały przez przeciętnego człowieka. To ostatnie założenie spowodowało, że w powstałym języku – nazwanym *Cobol* (***CO**mmon **B**usiness **O**riented **L**anguage*) – aż do przesady są wykorzystywane wyrazy z języka naturalnego (nawet popularne operacje arytmetyczne nie są oznaczane tradycyjnymi symbolami, ale słowami typu ADD lub MULTIPLY). Cobol także przeszedł różne ewolucje i przetrwał aż do dzisiejszych czasów.

Trzeba odnotować jeszcze jedno istotne zdarzenie z końca lat pięćdziesiątych ubiegłego wieku – w 1958 roku John McCarthy z MIT wymyślił abstrakcyjną notację do opisu funkcji rekurencyjnych, która łatwo dawała się rozszerzyć do języka programowania ukierunkowanego na przetwarzanie listowych struktur danych, nazwanego zatem *Lisp* (***LIS**t **P**rocessor*). Pod tym akronimem kryje się – podobnie jak to ma miejsce w wypadku języka Fortran – właściwie cała klasa języków, bo Lisp ulegał wielu modyfikacjom oraz pojawiały się jego najróżniejsze wersje i dialekty, takie jak *Lisp 1.5*, *Mlisp*, *Lisp-A*, *Interlisp*, *Common Lisp* czy *Scheme*. Tzw. *czysty Lisp* został zaimplementowany na komputerze IBM 704 (około 1960 roku) i – znowu podobnie jak w wypadku języka Fortran – znalazło to pewne odbicie w samym języku (operacje listowe *car* i *cdr*, to instrukcje z repertuaru komputera IBM 704). Natomiast pierwsze większe rozszerzenie języka, *Lisp 1.5*, zaimplementowano w 1962 roku na komputerze IBM 7090, a później na wielu innych.

2.3 JĘZYKI PROGRAMOWANIA A JĘZYKI SPECYFIKACJI

Zapis algorytmu w konkretnym języku programowania jest tylko jedną z faz procesu wytwarzania oprogramowania. Użycie ścisłej notacji jest niezbędne w całym tym procesie, poczynając od spisywania wyników analizy wymagań (co stanowi jego pierwszą fazę), a na kodzie konkretnego komputera skończywszy.

Rozważmy teraz bardzo krótko kwestię notacji używanej do specyfikacji na różnych poziomach abstrakcji, a dokładniej – związek między tą notacją a językami programowania.

Wspomniany wcześniej Algol 60 początkowo zaistniał tylko w wersji wzorcowej, którą powinno się traktować właśnie jako *język specyfikacji*. W wersji tej nie było w szczególności operacji wejścia-wyjścia, które pojawiały się dopiero w konkretnych realizacjach[5]. Te ostatnie mogły ponadto wprowadzać różne modyfikacje wersji wzorcowej (ograniczenia, rozszerzenia, zmiany składniowe etc.), po części niezbędnych przy opracowywaniu kompilatorów, a po części wynikających z inwencji twórców tychże. Przykładem bardzo udanej implementacji języka Algol 60 był Gier Algol w swoich kolejnych edycjach opracowany przez Petera Naura dla duńskiego komputera GIER. Z chwilą powstania konkretnych realizacji, język Algol ten stał się w pewnym sensie językiem *wykonywalnych specyfikacji*, gdyż przejście od specyfikacji do wykonywalnego kodu było już automatyczne, poprzez kompilację. Poprawność kodu względem specyfikacji zależała więc już tylko od poprawności tej kompilacji, co oznaczało konieczność (jednorazowego) zweryfikowania kompilatora.

Efektywność kodu wynikowego po kompilacji (zwłaszcza wobec nie zaawansowanych wówczas jeszcze technik tworzenia kompilatorów i przy kiepskich parametrach sprzętu, a jednocześnie wysokiej jego cenie) była przedmiotem dużej troski. Doprowadziło to do wypracowania możliwości włączania do programu w języku wysokiego poziomu wstawek napisanych w asemblerze. Innymi słowy została dana możliwość *łączenia* specyfikacji (tekst w języku wysokiego poziomu) z kodem (tekst w asemblerze) – techniki, która bywa i dzisiaj stosowana, choć na innym poziomie abstrakcji, i jest niekiedy niesłusznie uważana za nowatorskie rozwiązanie.

Widzimy więc, że rozróżnianie pomiędzy językiem specyfikacji a językiem programowania jest kwestią umowną.

[5] Autorzy języka Algol 60 uważali wówczas, że wprowadzanie danych nie wymaga specyfikacji, ma „charakter przyziemny" i powinno być uwzględniane dopiero przy opracowywaniu kompilatorów, a więc znajduje swoje odbicie w konkretnych realizacjach. Szybko jednak uznano to za błąd — następniki języka Algol 60 zajmowały się już wejściem-wyjściem na poziomie wzorcowym.

2.4 PODSTAWOWE PARADYGAMTY PROGRAMOWANIA

Przytoczone w poprzednich podrozdziałach przykłady wskazują w szczególności na metamorfozę komputerów i ich zastosowań. Z urządzeń elitarnych, dostępnych tylko dla wybrańców, zaczęły zwolna „trafiać pod strzechy" różnych instytucji, stawać się narzędziem pracy naukowców z wielu dziedzin, a także pojawiły się ich pierwsze praktyczne aplikacje – wynikało to oczywiście ze zmian technologicznych umożliwiających lepsze i tańsze konstrukcje sprzętowe. Zaowocowało to dalszym znacznym zwiększeniem zainteresowania językami programowania. Pojawiły się przy tym próby odmiennego podejścia do formułowania rozwiązywanych problemów.

Języki wewnętrzne komputerów, jak również wspomniane w poprzednim punkcie Fortran, Algol i Cobol, są nazywane *językami imperatywnymi*, ponieważ sposób formułowania w nich algorytmów sprowadza się do wydawania poleceń, typu: „rób po kolei to a to, a następnie – jeśli jest tak a tak – to kontynuuj, zaś w przeciwnym wypadku przejdź do innej, wskazanej fazy obliczeń". W językach tych mamy zmienne, którym są nadawane wartości w wyniku ewaluacji wyrażeń. W językach imperatywnych wysokiego poziomu występują też różne formy instrukcji warunkowej oraz iteracyjnej, a także mechanizm grupowania kilku instrukcji w jedną, złożoną. W zależności od języka mamy ponadto pewne możliwości definiowania abstrakcyjnych operacji (w postaci dość powszechnego aparatu procedur) oraz struktur danych (ograniczonych w początkowych językach do tablic, służących do reprezentowania macierzy). Języki te stanowią więc najbardziej naturalne „rozszerzenie" komputera.

Natomiast język Lisp (a raczej jego protoplasta *IPL – **I**nformation **P**rocessing **L**anguage*) zapoczątkował inny paradygmat programowania, zwany *funkcyjnym*, którego podstawą teoretyczną jest tzw. *rachunek lambda z typami*. Obliczenia polegają tutaj nie na wykonywaniu kolejnych instrukcji, ale na ewaluacji funkcji (często rekurencyjnych). Innymi słowy zapisujemy nie tyle *sposób obliczania* interesującej nas wartości, co *matematyczną funkcję* definiującą tę wartość. W czystym języku funkcyjnym znika zmienna i instrukcja imperatywna przypisania, a przy obliczaniu wartości funkcji nie mogą występować tzw. efekty uboczne. Najbardziej typowym przedstawicielem tej klasy jest język *Haskell* z końca lat osiemdziesiątych ubiegłego wieku (nazwany tak dla uczczenia amerykańskiego matematyka i logika, Haskella Curry'ego). Jego najnowsza wersja, to *Haskell 98*. Króluje jednak „mniej ortodoksyjne" podejście, stanowiące pewną fuzję stylu funkcyjnego z elementami programowania imperatywnego lub/i obiektowego. Reprezentantem tego nurtu jest wspomniany już wcześniej język Lisp z pochodnymi, a także *ML* (***M**eta **L**anguage*) i wywodzące się z niego *SML* (***S**tandard **ML***) oraz *Ocaml* (***O**bjective **CAML***).

Kolejny paradygmat, to *programowanie w logice*. Tutaj czołowym przedstawicielem jest język *Prolog* (***PRO**gramming **LOG**ic*), zaprojektowany w 1972 roku przez Alaina Colmerauera (francuskiego specjalistę od sztucznej inteligencji) do zastosowań przy przetwarzaniu języka naturalnego. Początkowo Prolog był znany i rozwijany tylko w niewielkim kręgu europejskich naukowców, by w początkach lat osiemdziesiątych stać się niemal światowym przebojem, gdy został nagłośniony japoński projekt „komputerów piątej generacji", w którym Prolog odgrywał kluczową rolę. Choć projekt ten nie okazał się sukcesem, paradygmat programowania w logice zyskał wielu gorących zwolenników, a Prolog stał się sztandarowym reprezentantem tego podejścia. W omawianym podejściu zapisujemy relacje i formuły logiczne, których prawdziwość jest weryfikowana w trakcie wykonywania programu. Programowanie w logice i programowanie funkcyjne mają na tyle dużo wspólnych cech, że pojawiają się języki, które z założenia należą do tych obu klas – koronnym przedstawicielem jest tu stosunkowo nowy język, *Mercury* (opracowany i rozwijany na Uniwersytecie w Melbourne).

Niezwykle ostatnio popularny paradygmat – *programowanie obiektowe* – ma swoje korzenie zarówno w licznych pracach związanych z modularyzacją, jak i w języku programowania *Simula 67*, którego twórcami byli norwescy naukowcy: Ole-Johan Dahl, Bjørn Myhrhaug i Krysten Nygaard. Trzeba jednak stwierdzić, że oryginalnie Simula stanowiła rozszerzenie języka Algol 60 o mechanizmy symulacji. Jednakże jest to pierwszy język, w którym pojawiają się typowe dla programowania obiektowego pojęcia, takie jak *obiekt* oraz *klasa*. Cechą charakterystyczną tego stylu programowania jest daleko idąca modularyzacja z „ukrywaniem sekretów" (zwana *hermetyzacją* lub *enkapsulacją*), grupowanie obiektów w *klasy* z możliwością *dziedziczenia* oraz *polimorfizm*. Pierwszym językiem jawnie projektowanym jako obiektowy był *Smalltalk*, opracowany w latach

siedemdziesiątych ubiegłego wieku w laboratorium naukowym firmy Xerox PARC, dzięki któremu pojawiły się tak powszechne obecnie rozwiązania, jak systemy okienkowe. Mamy też polski akcent – język Loglan (lata siedemdziesiąte i późniejsze; zespół Andrzeja Salwickiego z Uniwersytetu Warszawskiego) oraz cała gama innych języków, jak np. (alfabetycznie): *C++, C#, Delphi*, *Eiffel, Java, JavaScript, Oberon*, *Oclam, Perl*, *PHP*, *Python, Visual Basic*.

Ostatnim paradygmatem, który wszakże jest niekiedy wymieniany przy okazji innej klasyfikacji języków programowania, to *programowanie współbieżne*. Dotyczy ono sytuacji, w której algorytmy zapisujemy w postaci zbioru procesów, mogących ze sobą współpracować, ale konkurując przy tym o pewne zasoby. Mamy więc tutaj nie tylko klasyczne problemy do rozwiązania w ramach każdego z procesów, ale dodatkowo cały nowy bagaż kłopotów wynikających z konieczności synchronizacji współbieżnych procesów, zapewniającej ich bezpieczną współpracę i terminową realizację. Modelem wielu konkretnych rozwiązań językowych jest klasyczna propozycja Hoare'a z 1978 roku: *CSP* (***C****ommunicating* ***S****equential* ***P****rocesses*), na której bazował w szczególności język *Occam*. Do najbardziej znanych języków programowania współbieżnego należy niewątpliwie *Ada*, ale w klasie tej jest jeszcze wiele innych, gdyż praktycznie każdy nowoczesny język programowania ma jakieś mechanizmy do wyrażania współbieżności (podobnie jak i obiektowości).

Paradygmaty podzieliły więc języki programowania na kilka klas. Ale nie jest to jedyny możliwy podział. Często spotyka się na przykład klasyfikację ze względu na zastosowania: *języki do obliczeń naukowych* (np. Fortran), *języki do przetwarzania danych* (np. Cobol), *języki do programowania systemowego* (np. C), *języki czasu rzeczywistego* (np. Ada), *języki specjalistyczne* (np. Apt), *języki symulacyjne* (np. Simula 67) itd. Przy czym zarówno kryteria podziałów nie są na tyle ostre, aby klasy według różnych podziałów nie nachodziły na siebie, jak i są języki, które bądź od początku należą do paru klas (np. Ada pasuje do kilku z nich), bądź w miarę upływu czasu i powstawania nowych wersji nabierają nowych cech (np. dodawana obiektowość). Niektóre języki były projektowane od początku z myślą o kompilacji (np. język Pascal), a niektóre z myślą o interpretacji (np. Python) – te ostatnie nazywa się zwykle *językami interakcyjnymi* lub *konwersacyjnymi*.

Jak więc z tego widać mamy wiele różnych, często na siebie nachodzących klasyfikacji, a ponadto rzadko który język występuje w tak „czystej" postaci, że pasuje do dokładnie jednej klasy. Do tego trzeba pamiętać o ewolucji języków – wersja pierwotna i ta po kilkunastu lub kilkudziesięciu latach mają niekiedy niewiele, poza nazwą, wspólnego.

2.5 CZTERDZIEŚCI LAT MINĘŁO ...

Lata sześćdziesiąte i siedemdziesiąte ubiegłego stulecia, to istna eksplozja nowych języków programowania wysokiego poziomu wraz z różnymi próbami ich precyzyjnego opisu (por. artykuł Petera Landina z marca 1966 roku o znamiennym tytule The Next 700 Programming Languages, *Communications of the ACM* ***9***(3):157-166). Nim zajmiemy się Pascalem, który stanowił swego rodzaju fenomen, wspomnimy krótko o kilku innych ważnych – z różnego punktu widzenia – językach.

1. APL (***A*** ***P****rogramming* ***L****anguage*) – początkowo była to tylko specjalna notacja, opracowana około 1960 roku przez Kena Iversona z Harvardu, do zwięzłego opisywania algorytmów, w których występuje rachunek na macierzach. Z czasem przerodziła się w konwersacyjny język programowania o dość szerokich zastosowaniach, ale przy tym o bardzo udziwnionej i mało czytelnej składni. APL jest nadal rozwijany, mając zagorzałych zwolenników w niektórych środowiskach naukowych.
2. PL/I (***P****rogramming* ***L****anguage* ***I***) – język ten, opracowany przez George'a Radina z IBM w 1964 roku, miał być uniwersalnym „lekarstwem na wszystko" dzięki wykorzystaniu w nim najlepszych pomysłów z języków: Fortran, Algol 60 i Cobol. Co więcej, był to pierwszy w historii język z formalnie opisaną semantyką (przy użyciu notacji *VDL* – ***V****ienna* ***D****efinition* ***L****anguage*). Mimo wielu ciekawych rozwiązań PL/I nigdy nie uzyskał powszechnej akceptacji i nie zdobywszy popularności „zmarł śmiercią naturalną".
3. Basic (***B****eginner's* ***A****ll-purpose* ***S****ymbolic* ***I****nstruction* ***C****ode*) – pomyślany jako łatwy język dla nowicjuszy, do szybkiego nauczenia się prostego programowania. Był zaprojektowany przez Johna G. Kemeny i Thomasa E. Kurtza z Dartmouth College, a jego pierwsza realizacja – na komputerze ... IBM 704 – została ukończona w maju 1964. Wraz z pojawieniem się mikrokomputerów Basic stał się bardzo popularny i powszechnie używany, zwłaszcza przez dzieci i młodzież, choć jego walory dydaktyczne były często

podważane. Język ten nadal jest wykorzystywany, a dokładniej jego różne dialekty (mające coraz mniej wspólnego ze swoim protoplastą). Szczególnie popularna jest wersja obiektowa z 1987 roku, Visual Basic, stanowiąca zaawansowane i nowatorskie środowisko programistyczne.

4. Logo – to nie tylko język, ale cała filozofia podejścia do programowania przyjęta przez pedagogów szkolnych na całym świecie (grafika żółwia). Jej twórcą jest naukowiec z MIT, Seymour Papert, choć przy samym języku (lata 1966-1968) pracowało jeszcze kilka innych osób. Logo pozwala w sposób naturalny i intuicyjny przyswajać trudne i ważne pojęcia (jak np. procedury, rekurencję, strukturalne programowanie), przez co stało się naturalnym kandydatem do szkolnej nauki programowania. Język ten jest nadal popularny, choć na niższych niż dawniej etapach edukacji.
5. Algol 68 – język opracowany w latach 1965-68 (przez międzynarodowy zespół pod kierunkiem holenderskiego matematyka i informatyka Aada van Wijngaardena), który uzyskawszy „pieczątkę" Międzynarodowej Federacji Przetwarzania Informacji (IFIP) jako oficjalny następnik języka Algol 60, nie spełnił jednak oczekiwań i mimo różnych prób jego upowszechnienia zniknął z pola widzenia. Trzeba jednak lojalnie przyznać, że Algol 68 wniósł wiele do rozwoju metodyki programowania, języków programowania oraz metod ich opisu (por. np. gramatyki dwupoziomowe van Wijngaardena).
6. C – najkrócej mówiąc, jest to język opracowany przez programistów dla programistów. Był zainspirowany językiem *BCPL* (***B**asic **C**ombined **P**rogramming **L**anguage*) i powstał w roku 1972 w celu ponownego zaprogramowania systemu operacyjnego Unix (dla uzyskania przenośności tego systemu pomiędzy różnymi architekturami). Głównym twórcą języka C był Dennis Ritchie z firmy Bell Labs. Jak wiadomo, ten język osiągnął nadspodziewany sukces i jest do dzisiaj powszechnie stosowany na całym świecie, zarówno do programowania różnych aplikacji, jak i w dydaktyce (pomimo jego istotnych ułomności z tego punktu widzenia). Zapoczątkował on całą linię języków, w tym bardzo popularną wersję obiektową *C++* oraz szlagier ostatnich lat – *C#* (*C sharp*, czyli muzyczne *cis*).

2.6. ERA JĘZYKA PASCAL

W latach sześćdziesiątych zeszłego wieku, jak wspominaliśmy w poprzednim rozdziale, podjęto wiele prób opracowania nowego języka programowania licząc, iż powstanie produkt uniwersalny, powszechnie zaakceptowany. Niektóre z tych projektów miały przy tym wsparcie instytucjonalne (np. PL/I – firma IBM, Algol 68 – IFIP) oraz powstawały przy udziale zespołu specjalistów. I nie przyniosło to oczekiwanego rezultatu.

Tymczasem w 1971 roku szwajcarski informatyk Niklaus Wirth przedstawił w naukowym czasopiśmie nowy język programowania o nazwie *Pascal* (kontynuując zainicjowany przez siebie kilka lat wcześniej taki właśnie sposób honorowania wybitnych postaci historycznych). Język ten wyrósł z prac Wirtha nad następnikiem języka Algol 60 oraz uwzględniał najnowsze wówczas trendy w metodyce programowania. Prawie jednocześnie z tą publikacją powstała też w środowisku Wirtha pierwsza implementacja tego języka dla komputera CDC 6000.

Język Pascal oryginalnie był pomyślany jako notacja wspomagająca nauczanie programowania w systematyczny sposób. Związana z nim filozofia podejścia do programowania (np. unikanie instrukcji skoku, którą Wirth pozostawił wyłącznie „ze strachu", znając powszechne wówczas nawyki wśród programistów), prostota i klarowność języka oraz pojawienie się w nim zalążków abstrakcyjnych struktur danych, pozwalały wprowadzić nową jakość do metodyki programowania. Wkrótce okazało się, że Wirth „trafił w dziesiątkę" – język Pascal stał się niekwestionowanym standardem na całym świecie, który nie tylko dominował w środowisku akademickim, ale i sprawdzał się jako praktyczne narzędzie wykorzystywane do komercyjnych zastosowań. Efektywne kompilatory tego języka – a także towarzyszące im inne narzędzia wspomagające programowanie – były dostępne na praktycznie każdą architekturę, z mikrokomputerami włącznie. Pojawiały się nowe, ulepszone i rozszerzone wersje języka, w różnych paradygmatach programowania (by wspomnieć tylko *Concurrent Pascal, Simpascal, Borland Pascal*, *Delphi*).

Pomimo iż pojawiały się nowocześniejsze języki, a w szczególności sam Wirth przedstawił kilka kolejnych propozycji (*Modula, Modula-2*, *Oberon*), era języka Pascal trwała około 30 lat – dopiero teraz pojawiają się coraz widoczniejsze oznaki, iż czas ustąpić tronu.

3 DOKĄD ZDĄŻAMY

Pełniejsze rozważania na ten temat powinny być przedmiotem głębszych przemyśleń i oddzielnego opracowania. Poniżej zamieścimy więc tylko bardzo skrótowe refleksje, raczej sygnalizując trendy niż je omawiając.

3.1 JAVA, JAVA, JAVA

Język *Java* pojawił się w 1995 roku, pierwotnie pod nazwą *Oak*. Jego twórca – James Gosling z firmy Sun Microsystems – dobrze czując słabe punkty rosnącego w popularność języka C++ zaproponował jego dość istotne modyfikacje, obejmujące zarówno ograniczenia, jak i rozszerzenia. Chodziło o to, aby powstał przenośny język obiektowy, notacyjnie przypominający C++, w miarę prosty, o dużej sile wyrazu i nadający się do łatwego formułowania współcześnie rozwiązywanych problemów.

Doświadczenia ostatnich kilku lat wyraźnie wskazują na to, że ów ambitny cel został osiągnięty. Język Java i jego różne mutacje, wraz z ciągle wzbogacanym wspomagającym środowiskiem narzędziowym, zdobywa sobie coraz większą popularność, powoli przejmując kolejne obszary dotychczas okupowane przez inne języki i systemy. Na uwagę zasługuje w szczególności próba zdefiniowania specjalnego podzbioru tego języka dla celów edukacyjnych – znane amerykańskie towarzystwo naukowe ACM powołało zespół zadaniowy (ang. *task force*), który przygotował odpowiedni raport na ten temat[6]. Można oczekiwać, że jeśli wynik pracy tego zespołu zakończy się sukcesem, to Java – tak jak wcześniej język Pascal – stanie się światowym standardem w edukacji.

3.2 NOWE WYZWANIA

Jeszcze kilka lat temu mało kto przewidywał iż telefony komórkowe nie tylko staną się powszechne, ale i będą się scalały z innymi urządzeniami, tworząc swego rodzaju „multimedialne kombajny". To tylko jeden z przykładów świadczących o tym, że tempo rozwoju technologii teleinformatycznych przerosło nasze najśmielsze oczekiwania.

A za tym rozwojem kryją się oczywiście nowe potrzeby informatyczne i nowe wyzwania dla całego sztabu specjalistów z kolejnych faz procesu wytwórstwa oprogramowania, w tym programistów. Potrzebne są więc języki do wygodnego zapisywania innego rodzaju algorytmów, na innym poziomie abstrakcji, dotyczących nowych zastosowań i takie powstają – albo przez rozszerzenie tradycyjnych, albo jako nowe propozycje. Przykładem niech tu będą języki skryptowe (*PHP* lub *JavaScript*) do projektowania witryn internetowych, w tym tworzenia dynamicznych stron WWW – oczywiście kilkanaście lat temu nikt o takiej potrzebie nawet ***nie śnił***!

Natomiast pozornie tylko nowym wyzwaniem jest kwestia *komponentów*, czyli próby wielokrotnego wykorzystywania dobrze określonych i już sprawdzonych w praktyce „kawałków oprogramowania" oraz konstruowania systemów z takich gotowych elementów. Jeśli się chwilę zastanowimy, to dojdziemy do wniosku, że nie jest to ani nowa potrzeba, ani nowy pomysł. Każdy program jest przecież zbudowany z takich sprawdzonych kawałków – chodzi tylko o ich granulację: wielkość i funkcjonalność. Już biblioteki podprogramów (standardowych i niestandardowych) w pierwszych językach wysokiego poziomu stanowią dobre przykłady komponentów. Jeszcze wyraźniej to widać w koncepcji paradygmatu programowania obiektowego – modularyzacja z hermetyzacją służy właśnie temu, aby powstawały coraz bardziej wyspecjalizowane (ale i sparametryzowane) „cegiełki" do w miarę łatwego składania z nich większych systemów.

ZAKOŃCZENIE

Jak wspomnieliśmy na początku, problematyką języków zajmują się różne dziedziny naukowe, z językoznawstwem na czele. A w zakresie samych języków programowania, to problematyce ich ewolucji można poświęcić tomy – niniejsze opracowanie jest tylko krótkim wprowadzeniem do tej interesującej i obszernej tematyki.

Mamy jednak nadzieję, że Czytelnik podziela teraz w pełni myśl przewodnią:
Język językowi nie równy

[6] Por: http://www-cs-faculty.Stanford.edu/~eroberts/java/special-session.pdf

BIBLIOGRAFIA

[JM04] Jan Madey, Czy to sen, czy Java? O ewolucji języków programowania, w: M. M. Sysło (red.), Materiały Konferencji „Informatyka w Szkole XX", Wrocław 2004, str. 6-16, ISBN: 83-920799-2-2.

[DD72] David Diringer, *Alfabet czyli klucz do dziejów ludzkości*, PIW, Warszawa, 1972.

[SJP78] M. Szymczak (red. nauk.), *Słownik języka polskiego*, PWN, Warszawa 1978.

[WEPI] F. J. Granowski i S. J. Sikorski (wyd.), Wielka Encyklopedia Powszechna Ilustrowana, 1880 – 1914 (55 tomów, przerwana na haśle „Patroklos").

[WEP65] Wielka Encyklopedia Powszechna PWN, PWN, Warszawa 1965.

[WIEM] http://wiem.onet.pl/.

[Wikipedia] wolna encyklopedia: http://pl.wikipedia.org/wiki/.

Poza powyższymi pozycjami, do których są bezpośrednie odwołania, przy opracowywaniu tekstu korzystano jeszcze z bardzo wielu źródeł:

- stron internetowych wydawnictw encyklopedycznych;
- innych stron internetowych, w tym:
  – http://people.ku.edu/~nkinners/LangList/Extras/search.htm.
- różnych wydawnictw książkowych, w tym:
  –Michael Marcotty, Henry Ledgard: *The World of Programming Languages*, Springer-Verlag, 1987 (polskie wydanie: *W kręgu języków programowania*, WN-T, Warszawa 1991).
  – Ravi Sethi: *Programming Languages. Concepts and Constructs.* Second Edition, Addison-Wesley, 1996.
  – Oreste Vaccari, Enko Elisa Vaccari, *Pictorial Chinese-Japanese Characters*, Sixth Edition, Vaccari's Language Institute, Tokio, 1968.

W projekcie **Informatyka +**, poza wykładami i warsztatami, przewidziano następujące działania:

- 24-godzinne kursy dla uczniów w ramach modułów tematycznych
- 24-godzinne kursy metodyczne dla nauczycieli, przygotowujące do pracy z uczniem zdolnym
- nagrania 60 wykładów informatycznych, prowadzonych przez wybitnych specjalistów i nauczycieli akademickich
- konkursy dla uczniów, trzy w ciągu roku
- udział uczniów w pracach kół naukowych
- udział uczniów w konferencjach naukowych
- obozy wypoczynkowo-naukowe.

Szczegółowe informacje znajdują się na stronie projektu

www.informatykaplus.edu.pl