Standard Unicode 4.0. Wybrane pojęcia i terminy

Janusz S. Bień
Katedra Lingwistyki Formalnej UW

Streszczenie

Ukazanie się w sierpniu 2003 r. czwartej wersji standardu UNICODE (1504 strony plus CD-ROM), dostępnego także pod adresem `http://www.unicode.org`, stanowi okazję do zaprezentowania wybranych aspektów tego standardu. Unicode jest ukazany na tle wcześniej stosowanych metod kodowania tekstów. Artykuł zawiera również w sposób mniej lub bardziej jawny propozycje tłumaczenie używanych w standardzie angielskich terminów na język polski.

Wprowadzenie

Na wstępie warto przypomnieć różnicę między standardem a normą. Standardy – dawniej pisane i czytane *standarty*[1], ang. *standards* – to specyfikacje sformułowane przez pewne organizacje lub instytucje. Normy – ang. *legal standards* – to specyfikacje sformułowane przez krajowe lub międzynarodowe organizacje normalizacyjne, czyli przez instytucje upoważnione do tego przez państwo odpowiednimi ustawami; niektóre aspekty normalizacji krajowej i europejskiej omówiłem w artykule [2]. W skali światowej działalność normalizacyjna jest podstawowym źródłem utrzymania znaczącej liczby osób, stąd często można spotkać normy, których jedyną racją bytu jest dostarczanie zajęcia normalizatorom; standardy znacznie częściej zaspokajają rzeczywiste potrzeby ich użytkowników.

W [1] przedstawiłem krajowe, europejskie i międzynarodowe normy określające kodowanie tekstów polskich w systemach komputerowych. Tutaj podam tylko kilka przykładów kodowania polskich liter zgodnie z najnowszymi polskimi normami.

Według programu prac normalizacyjnych na r. 2004[2] w marcu br. powinne były zakończyć się prace nad normą PN-ISO/IEC 6937, będącą polskim tłumaczeniem normy ISO/IEC 6937:2001. Według tej normy napis `łąkę` kodujemy następująco (kody znaków podajemy w zapisie heksadecymalnym):

ł	ą		k	ę	
F8	BE	97	6B	BE	65
ł	˛	a	k	˛	e
1	2	3	4	5	6

Zgodnie z normą PN-ISO/IEC 8859-2:2001 – w praktyce równoważną odpowiedniemu fragmentowi standardu ECMA-94 ([7]) – i normą PN-I-10050:2002 ten sam napis kodujemy następująco:

	ł	ą	k	ę
PN-ISO/IEC 8859-2	B3	B1	6B	EA
PN-I-10050	7D	7B	6B	7C
	1	2	3	4

Żadna z wymienionych norm nie pozwala nam zakodować np. napisu ящик. Słowo to w izolacji można zakodować stosując np. międzynarodową normę ISO/IEC 8859-5:1999, w praktyce równoważną standardowi ECMA-113 ([8]):

я	щ	и	к
EF	E9	D8	DA
1	2	3	4

Jak jednak zakodować np. w tekście lingwistycznym napis «słowo ящик»?

Nie umożliwi tego polsko-europejska norma PN-EN 1923:2000 – ciągle jeszcze dostrzegalne są skutki żelaznej kurtyny – ale można zastosować normę PN-93/T-42118 razem z PN-ISO/IEC 2022:1996 lub PN-93/T-42112; te dwie ostatnie normy są w praktyce równoważne standardom ECMA-35 ([5]) i ECMA-48 ([6]). Polega to na użyciu tzw. sekwencji przełączających[3]. Jeśli założymy, że domyślnym kodowaniem jest opisany w normie PN-93/T42109.01 odpowiednik kodu ISO/IEC 646:1991 IRV – opisanego również w standardzie ECMA-6 ([4]) – czyli po prostu ASCII, to otrzymany rezultat przedstawia tabela na następnej stronie.

Choć ten sposób kodowania może wydawać się raczej niecodzienny, jest on stosowany m.in. w X-Windows przy kopiowaniu tekstu między applikacjami ([11]). Warto więc poświęcić mu chwilę uwagi.

[1] Patrz np. Ustawa z dnia 20 grudnia 1949 r. o utworzeniu Polskiego Komitetu Normalizacyjnego oraz o polskich normach i standartach (Dz.U. nr 63, poz. 403).

[2] Patrz `http://www.pkn.pl/PPN/ProgramPracKTnr170.htm`.

[3] Ang. *escape sequences*, dosłownie sekwencje ucieczkowe; *escape* bywa tłumaczone jako *unik*, ale określenie *sekwencje unikowe* nie brzmi najlepiej.

	s				ł	o	w	o					я	щ	и	к
	73	1B	2D	42	B3	6F	77	6F	20	1B	2D	4C	EF	E9	D8	DA
1	73															
2		1B	2D	42												
3		ESC	G1D6	04/02												
4					B3											
5						6F	77	6F	20							
6										1B	2D	4C				
7										ESC	G1D6	04/12				
8													EF	E9	D8	DA
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Zgodnie z przyjętymi w normach zasadami w 8-bitowym zestawie znaków wyróżnia się 2 obszary (lewy i prawy) znaków sterujących (CL i CR) oraz 2 obszary (lewy i prawy) znaków graficznych (GL i GR); ich interpretacja dokonuje się dwuetapowo. konkretną interpretację obszarów GL i GR oznaczamy odpowiednio G0 i G1. Założyliśmy, że obszar G0, obejmujący kody od 20 do 7F, zawiera kod ASCII, stąd litera `s` jest reprezentowana po prostu przez 73 (wiersz 1 w tabeli). Aby zakodować literę `ł`, zmieniamy odpowiednio interpretację G1. Czynimy to za pomocą sekwencji przełączającej (wiersze 2 i 3 w tabeli) składającej się z 3 znaków: znaku ucieczki, tzw. znaku pośredniego i znaku końcowego. Znak pośredni o akronimie G1D6 wskazuje, że zmiana interpretacji dotyczy G1, oraz określa pewne dodatkowe własności kodu. Bajt końcowy wskazuje natomiast na konkretny kod zarejestrowany w utrzymywanym na zlecenie Międzynarodowej Organizacji Normalizacyjnej ISO rejestrze *International Register of Coded Character Sets to be used with Escape Sequences*, dostępnym pod adresem `http://www.itscj.ipsj.or.jp/ISO-IR/`. Wartość 42 wskazuje konkretnie na kod ISO-IR 101, czyli kod zarejestrowany przez ISO pod numerem 101 o nazwie *Right-hand Part of Latin Alphabet No.2*. W tym kodzie litera `ł` reprezentowana jest jako B3 (wiersz 4 w tabeli). Ponieważ G0 nie uległo zmianie, pozostałe litery (wiersz 5 w tabeli) są kodowane jako ASCII. Aby jednak zakodować litery cyrylickie, musimy znowu zmienić interpretację obszaru G1 za pomocą odpowiedniej sekwencji przełączającej, tym razem z bajtem końcowym 4C wskazującym na ISO-IR 144 czyli *Cyrillic part of the Latin/Cyrillic Alphabet*.

Mam nadzieję, że powyższe przykłady unaoczniły niedogodności związane z kodowaniem tekstów za pomocą kodów jednobajtowych (o 7 lub 8 bitach).

Geneza i postać standardu Unicode

Aby uniknąć zilustrowanych wyżej problemów, Międzynarodowa Organizacja Normalizacyjna ISO pod koniec lat osiemdziesiątych ubiegłego wieku rozpoczęła prace nad normą ISO 10646 wprowadzającą wielobajtowy Uniwersalny Zestaw Znaków (*Universal Character Set* – UCS). Jak piszą Pike i Thompson ([9]), prace te były oparte na bardzo kontrowersyjnych założeniach. W szczególności kod miał być 4-bajtowy, a jego rozwój zdecentralizowany przez przydzielenie 16-bitowych fragmentów kodu krajowym organizacjom normalizacyjnych. Złożone głównie z amerykańskich firm komputerowych konsorcjum Unicode powstało – według Pike'a i Thompsona – jako protest wobec technicznych wad ISO 10646. Konsorcjum szybko przedstawiło swoją własną propozycję, która została opublikowana przez Addison-Wesley w 1991 r. jako standard Unicode 1.0. W tym samym roku rozpoczęły się negocjacje konsorcjum Unicode z odpowiednią grupą roboczą ISO, które doprowadziły do porzucenia przez ISO swojej pierwotnej koncepcji, oraz do dostosowania repertuaru znaków Unicode do ISO 10646. W rezultacie w 1993 ukazały się jednocześnie dwie publikacje: standard Unicode 1.1 i norma ISO 10646-1:1993, które opisywały ten sam repertuar znaków.

W sierpniu 2003 r. Addison Wesley opublikował *The Unicode Standard Version 4.0*. Wersja ta opisuje repertuar znaków zgodny z planowaną na ten sam rok publikacją normy ISO/IEC 10646, która jak dotąd (kwiecień 2004) nie doszła do skutku.

Numeracja wersji standardu jest trzypozycyjna. Pierwsza liczba oznacza wersję podstawową (ang. *major*), opublikowaną w książce wydanej tradycyjnie przez Addison-Wesley i dostępnej z pewnymi ograniczeniami na witrynie konsorcjum Unicode pod adresem `http://www.unicode.org`. Druga liczba oznacza wersję uzupełniająca (ang. *minor*), polegającą ma dodaniu dodatkowych znaków do wersji zasadniczej lub wprowadzeniu innych istotnych zmian, opublikowanych na witrynie konsorcjum jako raport techniczny.

Trzecia liczba, jeśli występuje, oznacza uaktualnienie normatywnych lub ważnych informacyjnych fragmentów standardu. Ostatnio takie uaktualnienie miało miejsce w marcu 2004 r. i aktualna wersja standardu powinna być – w języku angielskim – wskazywana w następujący sposób:

> The Unicode Consortium. The Unicode Standard, Version 4.0.1, defined by: The Unicode Standard, Version 4.0 (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), as amended by Unicode 4.0.1 (`http://www.unicode.org/versions/Unicode4.0.1/`).

Charakterystyczną cechą Unicode'u jest to, że nie ogranicza się on – jak norma ISO 10646 – do wyliczenia znaków i ich reprezentacji, ale opisuje ich własności i zasady przetwarzania. Co więcej, najważniejsze informacje dostępne są nie tylko w postaci słownego opisu, ale również w postaci plików komputerowych stanowiących integralną częśc standardu. Tak więc na wersję 4.0.1 składają się:

- Opis podstawowy: wspomniana wyżej książka opisująca wersję podstawową.
- Opis uzupełniający : brak.
- Opis uaktualnienia: wspomniana wyżej strona WWW.
- Aneksy – aktualnie 6, dostępne na witrynie Unicode, oznaczane UAX (*Unicode Standard Annexe*) z odpowiednim numerem[4]; tytuły podaję w swobodnym tłumaczeniu:

  UAX #9 Algorytm obsługi dwukierunkowości pisma (*The Bidirectional Algorithm*).

  UAX #11 Kategoria szerokości wschodnioazjatyckiej (*East Asian Width*).

  UAX #14 Łamanie wierszy (*Line Breaking Properties*).

  UAX #15 Normalizacja znaków (*Unicode Normalization Forms*).

  UAX #24 Nazewnictwo systemów pisma (*Script Names*).

  UAX #29 Segmentacja tekstu (*Text Boundaries*).
- Rejestr znaków Unicode'u (*Unicode Character Database* – UCD). Składa się on z kilkudziesiętu plików tekstowych (stąd moja niechęć do dosłownego tłumaczenia *database*) i kilku plików dokumentacyjnych w postaci czystego tekstu lub HTML. Pliki te są podzielone na następujące grupy:
  - Dokumentacja: 5 plików.
  - Informacje pierwotne (*Core Data*): 18 plików.
  - Informacje pochodne (*Derived Data*): 4 pliki.
  - Informacje wtórne (*Extracted Data*): 11 plików.
  - Dane testowe (*Test Data*): 1 plik.

  Nie będziemy oczywiście omawiać tutaj szczegółowo zawartości wszystkich tych plików.

[4] Zgodnie z amerykańskim zwyczajem numer oznaczany jest znakiem #.

Znaki piśmienne i ich kodowanie

Pojęcie znaku piśmiennego Podstawowym pojęciem standardu Unicode jest *abstract character*, co tłumaczę jako *znak piśmienny*, mając na myśli pierwotne znaczenie przymiotnika *piśmienny*, tj. dotyczący pisma. Definicja D3 ([12], s. 64) określa to pojęcie następująco:

> Jednostka informacji stosowana do organizacji danych tekstowych, sterowania nimi lub ich reprezentacji.

Definicja ta nawiązuje do definicji znaku (*character*) występującej od dawna w normach i podobnie jak one ma znaczenie czysto dekoracyjne – w żaden sposób nie pomaga ustalić, co jest znakiem piśmiennym w konkretnym przypadku.

Definicja D3 zawiera również dodatkowe wyjaśnienia:

- Znaki piśmienne służą do reprezentacji tekstu o charakterze symbolicznym, w przeciwieństwie do tekstów o charakterze wizualnym czy akustycznym. Chodzi więc m.in. o litery, ideografy, cyfry, znaki interpunkcyjne itp.
- Znak piśmienny nie ma konkretnego wyglądu i nie powinien być mylony z *glifem* (ang. *glyph*).
- Znak piśmienny niekoniecznie jest postrzegany przez użytkownika jako znak pisemny czy drukarski, w szczególności nie powinien być mylony z *grafemem.*
- Znaki piśmienne uwzględnione bezpośrednio w standardzie Unicode nazywamy *[podstawowymi] znakami Unicode'u* (ang. *Unicode abstract character*).
- Znak piśmienny nie będący znakiem Unicode'u często może być reprezentowany przez sekwencję znaków Unicode'u.

Wymienione wyżej kryteria nie są niestety ostre, dlatego najwygodniej jest przyjąć, że znaki piśmienne to pojęcie pierwotne, zdefiniowane przez wyliczenie. Ich zbiór nazywamy *repertuarem znaków piśmiennych* (ang. *abstract character repertoire* – ACR).

Znaki Unicode'u spełniające powyższe warunki to *znaki właściwe* (ang. *normal characters*, [12], s. 23). Jednak aby Unicode miał szansę przyjąć się w praktyce, niezbędne było zapewnienie pełnej zgodności z kodami będącymi w powszechnym użyciu. W konsekwencji w standardzie występują również znaki nie spełniające tych warunków, nazywane *znakami dostosowawczymi* (ang. *compatibility characters*).

W wersji 4.0 (i 4.0.1) Unicode'u repertuar znaków piśmiennych zawiera **96 382** znaki.

Znaki i współrzędne kodowe W celu przetwarzania znaków piśmiennych przez komputer przyporządkowuje się konkretnemu repertuarowi zestaw liczb całkowitych. Liczby te nazywamy *współrzędnymi kodowymi* (ang. *code points*) odpowiednich znaków, znak piśmienny tak reprezentowany nazywamy *znakiem kodowym* (ang. *encoded character*), a całe odwzorowanie *kodowym zestawem znaków* (ang. *coded character set* – CCS)[5].

Używane we wprowadzeniu słowo *kod* nie jest jednoznaczne, ale kontekst wyraźnie wskazuje, kiedy oznacza ono kodowy zestaw znaków, a kiedy konkretną współrzędną kodową.

Współrzędne kodowe muszą zawierać się w przestrzeni kodowej (ang. *codespace*), która dla Unicode'u w wersji 4.0 stanowi przedział – w zapisie heksadecymalnym – od 0 do 10FFFF. Przestrzeń ta może być powiększana w razie potrzeby.

Nie wszystkie współrzędne z tej przestrzeni są wykorzystane, a wśród wykorzystanych są takie, które nie reprezentują znaków piśmiennych, lecz mają inne funkcje. W związku z tym wyróżniamy następujące typy współrzędnych kodowych:

- Współrzędne znaków graficznych.
- Współrzędne znaków formatujących.
- Współrzędne znaków sterujących.
- Współrzędne użytku prywatnego (ang. *Private Usage Area* – PUA).
- Współrzędne surogatów (stosowane we wspomnianym niżej formacie kodowania UTF-16).
- Współrzędne nieznakowe (ang. *noncharacter*), zarezerwowane do użytku wewnętrznego.
- Współrzędne rezerwowe (ang. *reserved*), przewidziane do wykorzystania w przyszłości.

Współrzędne możemy klasyfikować również według ich wartości liczbowych. Wyróżniamy wówczas 4 *płaszczyzny*:

- Współrzędne od 0000 do FFFF: płaszczyzna 0 – *podstawowa płaszczyzna wielojęzyczna* (ang. *Basic Multilingual Plane* – BMP).
- Współrzędne od 1000 do 1FFFF: płaszczyzna 1 – *uzupełniająca płaszczyzna wielojęzyczna* (ang. *Supplementary Multilingual Plane* – SMP).
- Współrzędne od 2000 do 2FFFF: płaszczyzna 2 – *uzupełniająca płaszczyzna ideograficzna* (ang. *Supplementary Ideographic Plane* – SIP).
- Współrzędne od E000 do EFFFF: płaszczyzna 14 – *uzupełniająca płaszczyzna specjalna* (ang. *Supplementary Special-purpose Plane* – SSP).

Formaty kodowania znaków i jednostki kodowe Ze względu na budowę komputerów i urządzeń wejścia wyjścia liczby z pewnego przedziału są przetwarzane bardziej efektywnie niż inne – chodzi tutaj o liczby dające się zapisać w jednym bajcie, który w ogólnym wypadku może być 6-, 7- lub 8-bitowy. Takie jednostki pamięci czy transmisji nazywamy *jednostkami kodowymi* (ang. *code unit*).

Format kodowania znaków (to dość swobodne tłumaczenie ang. *character encoding form* – CEF) to odwzorowanie współrzędnych kodowych na zbiór sekwencji jednostek kodowych. W konkretnym formacie sekwencje jednostek kodowych mogą być stałej lub zmiennej długości.

W prostych przypadkach dla niewielkich repertuarów znaki piśmienne są odwzorowane na pojedyncze bajty 7-bitowe, jak w kodzie ASCII czy PN-I-10050:2002, lub 8-bitowe, jak w kodach PN-ISO/IEC 8859-2:2001 czy ISO/IEC 8859-5:1999. Kiedy jednak repertuar jest większy niż pojemność jednostki kodowej, jednym z możliwych rozwiązań jest stosowanie sekwencji o zmiennej długości; przykładem może być kodowanie PN-ISO/IEC 6937, w którym pewne znaki są reprezentowane przez jeden bajt, a inne przez dwa; warto zwrócić uwagę, że PN-ISO/IEC 6937 wyznacza repertuar znaków piśmiennych, ale nie wyznacza kodowego zestawu znaków w sensie zdefiniowanym wyżej.

W przypadku Unicode'u repertuar jest tak duży, że w użyciu jest kilka formatów kodowania. Najbardziej naturalny, ale również najmniej wygodny i efektywny jest zdefiniowany w ISO/IEC 10646 format **UCS-4** wykorzystyjący w pełni sekwencje czterech 8-bitowych bajtów, mogące reprezentować współrzędne kodowe z przedziału od 0 do 7FFFFFFF. Podobny do niego jest **UTF-32** (w tym wypadku UTF oznacza ang. *Unicode Encoding Form*, w innym może oznaczać *UCS Encoding format*), ale współrzędne kodowe muszą należeć do przedziału od 0 do 10FFFF. Analogicznie jest zdefiniowany **UCS-2**, który może być stosowany tylko do repertuatu podstawowej płaszczyzny wielojęzycznej.

[5] Nie są to jedyne możliwe ani powszechnie stosowane tłumaczenia angielskich terminów.

Znacznie większe znaczenia w praktyce mają formaty o zmiennej długości sekwencji jednostek kodowych. Należą do nich UTF-8 i UTF-16. **UTF-8** reprezentuje znak Unicode'u za pomocą od jednego do czterech 8-bitowych bajtów, które mają dodatkowo pewne wygodne własności. **UTF-16** wykorzystuje jedną lub dwie 16-bitowe jednostki kodowe (każda z dwóch jednostek, które wspólnie tworzą reprezentację pojedynczego znaku, jest nazywana surogatem).

Schematy kodowania znaków Wprowadzony dotąd terminy jeszcze nie wystarczają do opisania wszystkich spotykanych w praktyce aspektów kodowania znaków, stąd kolejne pojęcie – *schemat kodowania znaków* (ang. *character encoding scheme* – CES). Schemat taki może być prosty lub złożony.

Prosty schemat kodowania to po prostu omówiony wyżej format kodowania uzupełniony o informację, w jakiej kolejności są zapamiętywane lub transmitowane bajty składające się na jednostki kodowe i ich sekwencje.Tak więc dla formatu UTF-16 mamy schematy UTF-16BE i UTF-16LE, gdzie BE i LE znaczą odpowiednio *big-endian* i *little-endian*; analogicznie dla UTF-32 mamy UTF-32BE i UTF-32LE. Dla formatów korzystających z jednoelementowych sekwencji pojedynczych bajtów mamy natomiast do czynienia z trywialnym schematem nie różniącym się praktycznie od formatu kodowania.

Warto pamiętać, że prezentowany tutaj aparat pojęciowy nie powstał od razu, nowe pojęcia tworzone są sukcesywnie dla lepszego wyjaśnienia kwestii, które nie były rozstrzygnięte jednoznacznie lub z innych powodów sprawiały problemy w praktyce. Wynikiem tego ewolucyjnego podejścia jest fakt, że UTF-8, UTF-16 i UTF-32 mogą oznaczać zarówno formaty, jak i schematy kodowania znaków; dla uniknięcia nieporozumień można stosować odpowiednio oznaczenia UTF-16 CEF, UTF-32 CES itd. UTF-16 CES i UTF-32 CES są przykładami schematów złożonych.

Złożony schemat kodowania to taki, który wykorzystuje jeden lub więcej prostych schematów kodowania oraz jakiś mechanizm przełączania między nimi. Klasycznym przykładem jest m.in. zilustrowana we wprowadzeniu norma PN-ISO/IEC 2022:1996; schematy UTF-16 CES i UTF-32 CES różnią się natomiast od odpowiednich formatów tym, że kolejność bajtów jest wskazywana przez *znacznik kolejności bajtów* (ang. *Byte Order Mark* – BOM).

Bajtowe zestawy znaków i kodowanie transferowe Wprowadzone wyżej dość wyrafinowane pojęcia swoją rację bytu zawdzięczają subtelnościom normy ISO/IEC 10646 i standardu Unicode. Dla pełności obrazu warto powiedzieć, że w praktyce posługiwano się również pod różnymi nazwami pojęciem bezpośredniego odwzorowania repertuaru znaków na uporządkowane sekwencje bajtów. Dla takich odwzorowań, rejestrowanych przez Internet Assigned Numbers Authority (`http://www.iana.org/assignments/character-sets`), proponuje się obecnie nazwę *character maps* ([3]). Ponieważ nie widzę możliwości dosłownego tłumaczenia tego określenia, proponuję nazywać je po polsku *bajtowymi zestawami znaków.*

W wielu zastosowaniach – np. poczcie elektronicznej – pojawia się jeszcze jedno odwzorowanie, nazywane po angielsku *transfer encoding syntax.* Ponieważ słowo *syntax* w tej nazwie nie ma moim zdaniem uzasadnienia, proponuję nazwę tę tłumaczyć jako *kodowanie transferowe*; jego charakterystyczną cechą jest to, że kodowaniu podlegają napisy lub pliki, a nie poszczególne znaki. Jako swoisty rodzaj kodowania transferowego można traktować kompresję tekstu, a w szczególności metody specjalnie zaprojektowanego dla standardu Unicode, takie jak SCSU (*Standard Compression Scheme for Unicode*). Ciekawym przykładem kodowania transferowego jest również *Punycode*, stosowane w internacjonalizowanych nazwach domen internetowych ([10]).

Uwagi końcowe

W artykule tym skoncentrowałem się na mniej znanych i rzadziej opisywanych aspektach standardu. Pominięte kwestie, takie jak różnica między znakiem piśmiennym i glifem czy też sprawa unifikacji znaków i innych założeń projektowych standardu, są niewątpliwie ważne, ale są też omawiane w praktycznie każdym wprowadzeniu do Unicode'u.

Literatura

[1] Janusz S. Bień. Kodowanie tekstów polskich w systemach komputerowych. *Postscriptum* (ISSN 1427-0501) nr 27–29 (jesień '98 – wiosna '99), s. 4–27. `http://www.mimuw.edu.pl/~jsbien/publ/JSB-ktp98/`.

[2] Janusz S. Bień. O „pierwszym rozbiorze polszczyzny", Radzie Języka Polskiego i normalizacji. *Prace Filologiczne* XLVIII, s. 33-62, 2003. `http://www.mimuw.edu.pl/~jsbien/publ/JSB-PF03/`.

[3] Mark Davis. Character Mapping Markup Language. Unicode Technical Report #22, 2002-08-16. `http://www.unicode.org/reports/tr22/`.

[4] ECMA-6. *7-Bit Coded Character Set*, 6th edition. December 1991. `http://www.ecma-international.org/publications/standards/ECMA-006.HTM`.

[5] ECMA-35. *Character Code Structure and Extension Techniques*, 6th edition. December 1994. http://www.ecma-international.org/publications/standards/ECMA-035.HTM.

[6] ECMA-48. *8-Bit Coded Character Set Structure and Rules*, 3rd edition. December 1991. http://www.ecma-international.org/publications/standards/ECMA-048.HTM.

[7] ECMA-94. *8-Bit Single-Byte Coded Graphic Character Sets - Latin Alphabets No. 1 to No. 4*, 2nd edition. June 1986. http://www.ecma-international.org/publications/standards/ECMA-094.HTM.

[8] ECMA-113. *8-Bit Single-Byte Coded Graphic Character Sets - Latin/Cyrillic Alphabet*, 3rd edition. December 1999. http://www.ecma-international.org/publications/standards/ECMA-113.HTM.

[9] Rob Pike, Ken Thompson. Hello World or Καλημέρα κόσμε or こんにちは世界. http://plan9.bell-labs.com/sys/doc/utf.html (także utf.ps i utf.pdf).

[10] RFC 3492. Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA), A. Costello, March 2003, ftp://ftp.rfc-editor.org/in-notes/rfc3492.txt.

[11] Robert W. Scheifler. Compound Text Encoding. Version 1.1.xf86.1, XFree86 4.0.2., XFree Inc. http://www.xfree86.org/current/ctext.html.

[12] The Unicode Consortium. *The Unicode Standard, Version 4.0*. Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1.

[13] Ken Whistler, Mark Davis, Asmus Freytag. Character Encoding Model. Unicode Technical Report #17 (Proposed Update, 2004-03-25). http://www.unicode.org/unicode/reports/tr17/tr17-4.html.