KODOWANIE NIEJEDNO MA IMIĘ. SPOSOBY REPREZENTOWANIA INFORMACJI

Paweł Perekietka
V Liceum Ogólnokształcące
im. Klaudyny Potockiej w Poznaniu
pawel.perekietka@gmail.com

Abstract. The word "code" is often used to talk about programming. However in this article the sense of coding that will be used is about clever representations of information. The article recalls the principle of parallelism in didactics: the individual learning process should follow historical order.

> To nie może być nasz alfabet.
> 360° i 26 znaków plus karta pytań to 13° na jedno pole.
> Stanowczo za mało. Nie sposób określić, co wskazuje obiektyw…
> Już wiem! Pomoże system szesnastkowy!
>
> [Mark, główny bohater powieści Marsjanin Andy'ego Weira]

1. Wstęp

Prostą konsekwencją postępującego przez wiele lat rozmywania znaczenia terminu *informatyka* jest dość powszechne przekonanie o tym, że to dziedzina wyłącznie praktyczna, a więc pozbawiona pojęć i metod naukowych. Taki uproszczony obraz informatyki (w wersji *pop*) dziś dominuje, oczywiście również w szkole.

Dziś, gdy jako główny cel powszechnego kształcenia informatycznego do podstawy programowej wprowadza się kształcenie myślenia komputacyjnego (ang. *computational thinking*)[1], a więc również kształcenie umiejętności programowania, istnieje ryzyko zawłaszczenia innego terminu, jakim jest *kodowanie*.

[1] We wprowadzeniu do rozporządzenia MEN o nowej podstawie programowej 14 lutego 2017 r. znajdujemy następujące objaśnienie istoty umiejętności myślenia komputacyjnego (informatycznego): „Obejmuje szeroki zakres intelektualnych metod i narzędzi, mających swoje źródło w informatyce, wywodzących się z komputerowego przetwarzania informacji i rozwiązywania problemów z pomocą komputerów w różnych dziedzinach. Integruje ludzkie myślenie z możliwościami komputerów."

Termin ten w dydaktyce ostatnich lat stał się słowem-kluczem na określenie działań popularyzujących naukę programowania, nie tylko w Polsce[2].

Naukowe znacznie terminu *kodowanie* w informatyce jest inne: *kodowanie to przekształcanie informacji w ciąg znaków lub cyfr*.[3] Celem kodowania jest znalezienie sposobu reprezentacji informacji dla jej zapisu lub przesłania do odbiorcy[4].

Autorzy podstawy programowej pierwszy cel kształcenia informatycznego określają tak: *Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i* ***sposobów reprezentowania informacji*** **(**podkreśl. PP). Podstawa programowa dla szkół podstawowych formułuje *explicite* jedno wymaganie szczegółowe (dla klas VII-VIII): „Uczeń przedstawia sposoby reprezentowania w komputerze wartości logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów".[5]

Ten artykuł ma za cel ukazanie prostych, choć niebanalnych, przykładów, wartościowych dla dydaktyki informatyki w szkole podstawowej (zwłaszcza w klasach VII i VIII), ilustrujących pojęcia *kodu* i *cyfryzacji*. Są to zarówno przykłady rozwiązań współcześnie używanych, jak i przykłady z historii szeroko pojętej teleinformatyki i techniki komputerowej. W przypisach dodano odnośniki do źródeł, anegdoty, dodatkowe komentarze merytoryczno-metodyczne i propozycje rozwinięcia tematu[6].

---

[2] Twórca projektu *code.org* wybór słowa-klucza uzasadniali prosto: krótkie i chwytliwe. Więcej na temat założeń tej inicjatywy można przeczytać w *Code.org's Educational Philosophy.*

[3] W *Słowniku języka polskiego PWN* termin *kodować* objaśnia się: „przetwarzać informacje na kod, czyli umowny system znaków umożliwiający przetworzenie informacji oraz odczytanie jej przez osoby znające ten system lub przez urządzenia pracujące w tym systemie".

[4] Mamy też problem z uproszczeniem znaczeniowym innego terminu: *cyfryzacja*. Niektórzy wskazują, że źródłem problemu jest to, że w języku polskim brakuje odpowiedników dla terminów języka angielskiego: 1. *digitization: the action or process of digitizing; the conversion of analogue data into digital form* 2. *digitalization: the adoption or increase in use of digital technology by an organization, industry*, etc."W języku polskim takiego rozróżnienia nie ma, terminy cyfryzacja i digitalizacja, stanowią synonimy. Por. sjp.pl/cyfryzacja

[5] Zaskakuje, że na liście nie wymieniono grafiki. Szkoda, że podstawa programowa wprost nie wskazuje innych propozycji: prostych kodów korekcyjnych czy metod kompresji. Wydaje się, że autorzy programów nauczania mogą twórczo ukonkretnić inne wymaganie szczegółowe *Uczeń prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów* posługując się właśnie przykładami prostych pojęć i algorytmów kodowania i kompresji.

[6] Wydaje się, że do tej pory w języku polskim nie powstało żadne opracowanie na temat sposobów reprezentacji informacji, napisane z myślą o nauczycielu szkoły podstawowej

2. Pojęcie kodu. Istota kodowania

By móc zapisać jakąś informację, trzeba stworzyć dla niej *kod*, to jest przedstawić ją w postaci specjalnych znaków albo sygnałów: impulsów elektrycznych, świetlnych, dźwiękowych itp. Z kodowaniem spotykamy się co krok. Nie tylko kiedy zastępujemy tekst symbolami lub przekładamy z jednego języka na drugi.

Jest wiele ciekawych przykładów kodów, zwanych czasami kodami użytkowymi, którymi można posłużyć się w szkole do ilustracji pojęcia kodu. Oto kilka z nich[7]:

- numery linii komunikacji miejskiej[8],
- tablice rejestracyjne samochodów,
- numer konta bankowego IBAN i numer karty kredytowej,
- numery PESEL[9] i numery NIP,
- kody pocztowe (ZIP) i kody paskowe PostNET,
- kody kreskowe[10] (np. EAN-13) i kody QR.

2.1. Pojęcie kodu dwójkowego (binarnego)

Do najdogodniejszych i najoszczędniejszych należy kod dwójkowy (binarny), zwany również zerojedynkowym, powszechnie stosowany w technice komputerowej i telekomunikacyjnej.

---

lub gimnazjum. Dla samokształcenia nauczyciela polecić można klasyczny podręcznik dla uczniów szkół ponadgimnazjalnych [15] i towarzyszący mu przewodnik metodyczny.

[7] Temat kodów użytkowych to dobry temat na zadanie domowe (projektowe). Szukanie informacji na temat czasu i miejsca powstania kodów, ich twórców, celu ich wprowadzenia, zasad działania (i modyfikacji później wprowadzonych) oraz dzisiejszych zastosowań może być dla niektórych pasjonującym i pouczającym interdyscyplinarnym wyzwaniem.

[8] Konkretny numer linii (np. 20) odpowiada określonej trasie.

[9] Przy okazji pojawi się wątek cyfry kontrolnej.

[10] To kody binarne. Każda cyfra jest reprezentowana przez pewną liczbę ciemnych i jasnych elementów (kresek). Czytnik kodu kreskowego emituje światło, które zostaje odbite od elementów jasnych, zaś pochłaniają je elementy ciemne. Światło odbite wraca do czytnika i powoduje powstanie w nim słabych sygnałów elektrycznych, światło pochłonięte przez kreski ciemne wywołuje sygnały silniejsze – podobnie jak w kodzie Morse'a. W zależności od grubości kresek, różna jest też długość trwania poszczególnych sygnałów. Powstaje ciąg sygnałów elektrycznych o różnym natężeniu i różnej długości. Otrzymane w ten sposób impulsy są tłumaczone przez dekoder czytnika na kod ASCII. W tym formacie dane trafiają do komputera. Pomysłodawca kodu paskowego, Joe Woodland, wspominał, że przeżył „oświecenie", gdy odpoczywał… na plaży. Jako były skaut znał dobrze kod Morse'a. Jego pomysł kodu paskowego stanowił jakby „przedłużenie" kodów Morse'a do drugiego wymiaru. Kody pierwotnie miały postać współkoncentrycznych pierścieni.

W obrazowy sposób pojęcie kodu, a w szczególności kodu binarnego (w kontekście komputerowej reprezentacji litery) wyjaśnia autor [9]: „Kod to po prostu coś, co zastępuje coś innego. Kodujemy dlatego, że kod jest bardziej funkcjonalny od rzeczy którą koduje. W przypadku kodowanego pisma – bo pozwala zatrudnić maszynę. Jeśli mówię A, to wprost słychać A. Gdy nie można mnie usłyszeć – napiszę A. Pozbawionej zmysłów maszynie mogę podać szereg bitów: 01100001[11]. Albo niech je sama odczyta z liczby 97 (gdy jej wartość przetłumaczy sobie na system dwójkowy). Jeszcze prościej będzie, gdy sama odczyta liczbę 97 i zrobi resztę, gdy ja tylko wcisnę klawisz A."

W zrozumieniu koncepcji kodu binarnego pomóc może przyjrzenie się alfabetowi Braille'a, który umożliwia odczytywanie tekstów osobom niewidomym i niedowidzącym. Ponad 200 lat temu 15-letni Francuz Louis Braille (1809-1852) wymyślił[12] system zapisu tekstu (liter, cyfr, znaków przestankowych itd.) z użyciem tzw. sześciopunktów jako kombinację sześciu wypukłych punktów ułożonych w dwóch kolumnach po trzy punkty w każdej:

System stał się bardzo popularny wśród osób niewidomych, gdyż umożliwił względnie szybki i niezawodny sposób „czytania" tekstu.

Obecnie, na mocy dyrektywy Unii Europejskiej obowiązującą normą stało się etykietowanie opakowań leków za pomocą alfabetu Braille'a. Coraz częściej można spotkać tę formę znakowania również na kartonikach dedykowanych do sektora kosmetycznego, chemii użytkowej, a nawet na opakowaniach spożywczych[13].

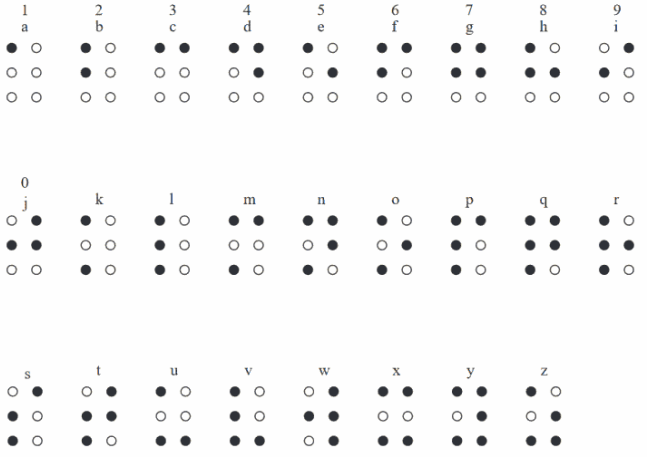
---

[11] Należy pamiętać, że sformułowanie „komputer zapisuje informacje w postaci ciągu 0 i 1" jest skrótem myślowym. Nie ma sposobu, by zrobić to bezpośrednio – komputery posługują się własnościami fizycznymi takimi jak np. wysokie i niskie napięcie. W przypadku człowieka, przynajmniej Polaka, można by sobie wyobrazić reprezentację binarną jako rozróżnienie np. kiwania głową („tak", czyli 1) i potrząsania głową („nie", czyli 0).

[12] Pomysł Braille'a powstał na bazie pisma, które opracował kapitan Charles Barbier, aby ułatwić żołnierzom armii napoleońskiej porozumiewanie się bez słów w ciemności. Braille zredukował liczbę kropek i dzięki temu każdą literę i cyfrę łatwiej rozpoznać palcem. Historia wynalazku (wraz z opisem trudności i braku zrozumienia, jakie napotkał Braille) jest przedstawiona w opowiadaniu pt. *Światło w ciemnościach* w książce [13, s. 201-206]. Z pewnością na podstawie opowiadania zainteresowani uczniowie mogliby, w porozumieniu z nauczycielem, przygotować inscenizację na zajęcia (nauczanie wyprzedzające).

[13] To oznacza, że łatwo dostępny jest prosty środek dydaktyczny.

Alfabet Braille'a można uznać za jeden z pierwszych przykładów „binarnego" zapisu informacji – używa się w nim bowiem tylko dwóch znaków (wypukły punkt lub jego brak).



Kody Braille'a dla liter i cyfr

W obrębie sześciopunktu można uzyskać 63 wzorce (znaki)[14], służące do oznaczania całego alfabetu oraz znaków interpunkcyjnych oraz znaków specjalnych (np. znak dużej litery, znak liczbowy).

Alfabet Braille'a jest dobrą ilustracją przyczyn popularności binarnego zapisu informacji. Można sobie wyobrazić system używający trzech rodzajów kropek: niewypukłych, półwypukłych i wypukłych. Kłopot w tym, że potrzebne byłyby bardziej dokładne urządzenia do tworzenia wypukłych kropek, a czytający musieliby więcej uwagi poświęcać rozróżnianiu kropek. Wystarczyłoby wtedy, że kartka zostałaby przygnieciona, nawet bardzo nieznacznie, a informacja stałby się nieczytelna.

2.2. Przykłady kodów cyfrowych

Kod Polibiusza

Zasadę kodowania cyfrowego, nie rozumianego jako szyfrowanie, zastosował po raz pierwszy prawdopodobnie grecki matematyk Polibiusz (II w. p.n.e.), który zakodował 25 liter alfabetu przez 25 par pięciu cyfr.

[14] Niebanalnym wyzwaniem zajęć może być sformułowanie przekonującego uzasadnienia. Takie zadanie proponują m.in. autorzy nowozelandzkiego szkolnego przewodnika po informatyce w rozdziale *Data representation*, gdy podają przykładu alfabetu Braille'a [12].

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Szachownica Polibiusza

Polibiusz uważał, że jego system kodowania może posłużyć jako pionierski sposób wysyłania wiadomości z latarni morskich za pomocą świateł błyskowych.[15]

Pomysł Polibiusza był przez wieki stosowany jako element składowy różnych metod szyfrowania (utajniania) korespondencji, zwłaszcza w średniowieczu.[16]

Kod Bacona

Pierwszym, który spostrzegł, że do jednoznacznego kodowania wystarczą tylko dwie cyfry był prawdopodobnie Francis Bacon (1561-1626), angielski dworzanin[17]. Był to kod, który dziś nazwalibyśmy kodem 5-bitowym[18].

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I, J**	**K**	**L**	**M**
aaaaa	aaaab	aaaba	aaabb	aabaa	aabab	aabba	aabbb	abaaa	abaab	ababa	ababb
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U, V**	**W**	**X**	**Y**	**Z**
abbaa	abbab	abbba	abbbb	baaaa	baaab	baaba	baabb	babaa	babab	babba	babbb

Warto zauważyć, że słowa kodu są uporządkowane leksykograficznie[19].

---

[15] W [10, s. 78] czytamy: „System [Polibiusza] był często wykorzystywany przez więźniów, zwłaszcza anarchistów więzionych w carskiej Rosji oraz jeńców wojennych w czasie wojny w Wietnamie. Porozumiewali się oni, wystukując poszczególne sekwencje liczb. Kod nie służył do ukrycia wiadomości, lecz stanowił prosty środek komunikacji między celami". Akronim IWE zapisany w kodzie Polibiusza wygląda następująco: 24 52 15.

[16] Jeszcze podczas I wojny światowej niemieccy wojskowi stosowali szyfr ADFGX (i modyfikację ADFGVX), w którym do kodowania wstępnego wiadomości używali szachownicy Polibiusza. Rzędy i kolumny szachownicy Polibiusza twórcy szyfru ADFGX opisali nie kolejnymi cyframi 1, 2, 3, 4 i 5 a literami A, D, F, G i X. Warto zauważyć, że te litery zostały wybrane jako najbardziej charakterystyczne podczas transmisji z użyciem kodu Morse'a (w tym sensie, że ich kody znacząco różniły się). Miało to na celu zmniejszenie ryzyka powstania błędów przy nadawaniu lub odbiorze wiadomości.
Przykład szyfru ADFGX można to w szkole użyć do wyjaśnienia różnicy między kodowaniem a szyfrowaniem (najpierw kodujemy, później szyfrujemy).

[17] Kod Bacona był elementem systemu szyfrowania sekretnych wiadomości.

[18] Kod stosowany przez Bacona znalazł ponowne zastosowanie w telegrafie Baudota.

[19] Dobrze, by uczniowie to sprawdzili samodzielnie. O ile wcześniej sami tego nie zauważyli.

Kod Morse'a

Samuel F. B. Morse (1791-1872) opracował (współpracując m.in. z Alfredem Vailem) sposób kodowania liter oraz cyfr oraz ich zamiany w impulsy elektryczne[20]. Kod opiera się na binarnym systemie kropek i kresek: kropka to jeden krótki sygnał, natomiast kreska to pojedynczy sygnał trwający tyle, co trzy kropki[21].

A	• —	N	— •
B	— • • •	O	— — —
C	— • — •	P	• — — •
D	— • •	Q	— — • —
E	•	R	• — •
F	• • — •	S	• • •
G	— — •	T	—
H	• • • •	U	• • —
I	• •	V	• • • —
J	• — — —	W	• — —
K	— • —	X	— • • —
L	• — • •	Y	— • — —
M	— —	Z	— — • •

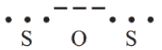
Metoda Morse'a bardzo szybko zyskała światowe uznanie i pozostała jednym ze standardowych systemów kodowania binarnego[22] aż do początku XXI w.

[20] Choć kod był zaprojektowany z myślą o telegrafie, można go było używać do komunikacji na odległość za pomocą innych środków, takich jak syreny okrętowe czy lampy błyskowe.

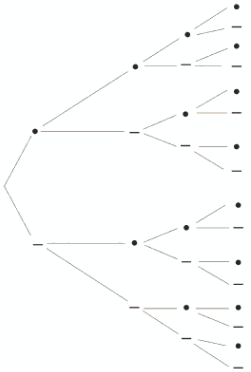
[21] Pierwotnie (1837 r.) Morse przesyłał tylko cyfry, z których za pomocą księgi kodowej można było odtwarzać słowa. Dopiero ok. 1840 r., w celu oszczędności baterii, zastosował kod o zmiennej długości. Zmagania Morse'a w ciekawy sposób opisuje autor książki [6]. W czasie zajęć z uczniami warto, by nauczyciel podkreślił, że genialne odkrycia i projekty wynalazków to często długi proces twórczy i walka z przeciwnościami.

[22] Precyzyjnie mówiąc w kodzie Morse'a występuje jeszcze trzeci znak, mianowicie *przerwa* między dwoma słowami. Przez przyporządkowanie:● (01), ─ (011) i *przerwa* (000) można kod Morse'a przekształcić w kod dwójkowy w ścisłym znaczeniu tego słowa. Takie przyporządkowanie było najczęściej podstawą dla technicznej realizacji kodu Morse'a (formułuje się je zazwyczaj w postaci reguły: „Kropka ma długość pauzy, kreska ma długość trzech kropek, przerwa ma długość trzech pauz", gdyż łatwo je było zrealizować przez stany fizyczne *prąd włączony* i *prąd wyłączony*. [2, s. 44] Już w roku 1841 Wheatstone zbudował nadajnik telegraficzny ze sterowaną taśmą dziurkowaną, używając dwójkowego (binarnego) przedstawienia znaków Morse'a.

W 1908 roku wprowadzono międzynarodowy sygnał wezwania o pomoc[23]:



Kod Morse'a można przedstawić jako drzewo kodowe[24].



Ten sposób prezentacji pozwala lepiej objaśnić, dlaczego kody przypisane różnym literom alfabetu są różnej długości: chodziło o efektywność kodu. Morse literom najczęściej występującym w tekstach języka angielskiego (e i t) przydzielił najkrótsze kody (odpowiednio: kropkę i kreskę).[25]

Kody telegraficzne

Rozwój telegrafii z XIX w. bez wątpienia był ściśle związany z rozwojem handlu międzynarodowego. Opłaty za telegramy były uzależnione od liczby liter i przedsiębiorcy szybko zorientowali się, że przesyłanie skróconych komunikatów pozwoli zaoszczędzić pieniądze. Dzięki specjalnym kodom firmy mogły tworzyć kombinacje znaków dla słów i wyrażeń często używanych w ich branży, a także szyfrować wiadomości do celów bezpieczeństwa[26]. Np. *kod Bentleya* zawierał takie kombinacje znaków, jak ATGAM ("have they authorised?", czyli „czy są upoważnieni?")[27]

---

[23] Sygnały SOS wysyłał radiooperator Titanica. Udało się ocalić 700 z 2200 osób, gdyż statki ratownicze przypłynęły zbyt późno. Tragedii można było zapobiec, gdyby kapitan statku nie ignorował otrzymanych telegraficznie ostrzeżeń o dryfujących górach lodowych.

[24] Zajęcia o kodzie Morse'a to znakomita okazja, by wprowadzić pojęcie *drzewa binarnego*.

[25] Morse'a wypada uznać za ojca kompresji. Por. [16]

[26] Średnia liczba liter w słowie języka angielskiego to 4,5. Z tej wiedzy korzystały firmy telegraficzne i zaczęły naliczać opłaty na podstawie liczby słów (a nie liter). To oznaczało m.in. konieczność stosowania 5-znakowych grup znaków w szyfrowanej korespondencji.

[27] Warto w czasie zajęć z uczniami przywołać określenie *w telegraficznym skrócie*.

Jedną ze stosowanych metod szyfrowania była metoda *Morbit*[28]:

1. Zapisywano tekst jawny w kodzie Morse'a (przerwy między literami kodowano znakiem */*, a przerwy między wyrazami – dwoma znakami *//*)[29] .
2. Następnie grupowano znaki (symbole) po dwa.
3. Wreszcie następowało właściwe szyfrowanie (podstawieniowe) z użyciem klucza[30] znanego nadawcy i odbiorcy wiadomości.
4. Na końcu cyfry szyfrogramu grupowano po pięć.

Np. dla tekstu jawnego NINETEEN i klucza

..	.—	./	—.	— —	—/	/.	/—	//
2	7	4	1	9	3	5	6	8

otrzymamy kolejno:

—./../—././—/././—.

/— ./ .. /— ./ ./ —/ ./ ./ —.

64264 43441

Kod Baudota-Murray'a

Zmienna długość kodów Morse'a była źródłem trudności przy konstrukcji automatycznych urządzeń telekomunikacyjnych, które pracowałyby w tym kodzie. Dlatego już w końcu XIX w. zaczęto opracowywać inne kody, w których znaki alfanumeryczne były kodowane za pomocą impulsów o tej samej, stałej liczbie elementów, tj. impulsów i przerw między nimi. Jednym z nich był kod, który stworzył francuski inżynier Emilie Baudot (1845-1903). Wszystkie kody miały tę samą długość pięciu symboli[31], co znacznie ułatwiało kodowanie i dekodowanie[32].

---

[28] Metoda *Morbit* jest przykładem kodowania wielowarstwowego. Z dydaktycznego punktu widzenia pozwala objaśnienie różnicy między kodowaniem i szyfrowaniem. W dzisiejszym programowaniu komputerowym idea działania kodów wielowarstwowych jest analogiczna.

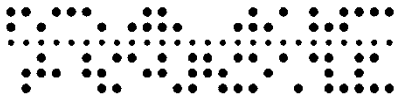
[29] Zwykle kod tekstu jawnego poprzedzano dodatkowym znakiem */*. Na końcu dodawano się znak */*, jeśli była taka konieczność (tak, aby łączna liczba symboli była parzysta).

[30] Klucz był permutacją cyfr 1, 2, 3, …, 9.

[31] W telegrafii stosowano wyłącznie duże litery. Nie było osobnych kodów dla cyfr. Istniał kod *przełączający* między podzestawem znaków i cyfr.

[32] Kody wprowadzano dwoma palcami lewej i trzema palcami prawej ręki za pomocą klawiszy przypominających fortepianowe. Więcej informacji o historii tych kodów patrz [9].

W latach 20. i 30. XX w. zaczęto budować międzynarodową sieć dalekopisową i teleksową, zgodnie z pewną modyfikacją kodu Baudota wprowadzoną na przełomie wieków XIX i XX przez Nowozelandczyka Donalda Murray'a (1865-1945). Kod Murray'a był dobrany tak, by litery najczęściej używane wymagały najmniejszej aktywności maszyny, na przykład zakodowanie litery E wymagało wykonania tylko jednej dziurki w taśmie telegraficznej[33].



Litery a, b, c, …, z przedstawione na taśmie papierowej

Każdemu znakowi odpowiada w ilustracji powyżej kombinacja pięciu otworów lub niewybitych prostych miejsc na pasku taśmy telegraficznej, rozdzielona przez rząd drobnych dziurek umożliwiających przesuwanie taśmy w maszynie[34].

3. Kody w komputerze

Kodowanie i dekodowanie to ważne czynności w ciągu czynności tworzących proces komputerowego przetwarzania danych[35]. Cały ten proces zaczyna się zazwyczaj od przekodowania danych z kodu dostępnego dla człowieka na odpowiedni kod maszynowy (w komputerze), kończy się zaś przekodowaniem danych zapisanych w sposób dostępny dla komputera na zapis odczytywalny przez człowieka.

Współczesne komputery przechowują informacje zakodowane binarnie, w postaci ciągów bitów, które w różnym kontekście mogą mieć bardzo różne znaczenie.

[33] Miało to znaczenie: maszyna dla przekazania tekstu jednej strony maszynopisu wykonywała ok. 10 tysięcy dziurek, co wymagało konserwacji jej licznych części mechanicznych.

[34] Podczas II wojny światowej (od 1941 r.) do utajniania niemieckiej korespondencji na najwyższym szczeblu (np. między kwaterą główną Hitlera a sztabami generalnymi rozsianymi po całym świecie) wykorzystywano maszynę szyfrującą skonstruowaną w firmie Lorentz. Tekst jawny wybijano w kodzie Baudota na taśmie perforowanej. Przy wczytywaniu czytnik elektryczny sprawdzał każdy z pięciu rzędów otworów. Wynik skomplikowanego szyfrowania zapisywano na nowej taśmie. Więcej w [8, s. 237-240].

[35] Inne czynności procesu przetwarzania danych to to: działania arytmetyczne, działania logiczne, porządkowanie zbiorów danych i transmisja danych, por. [14, s. 7-8].

```
01000010 01001101 01001110 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00111110 00000000 00000000 00000000 00101000 00000000
00000000 00000000 00100000 00000000 00000000 00000000 00000100 00000000
00000000 00000000 00000001 00000000 00000001 00000000 00000000 00000000
00000000 00000000 00010000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 11111111 11111111 11111111 00000000 11111111 11111111
11111111 11111111 11001100 11001100 11001100 11001100 10101010 10101010
10101010 10101010 11111111 11111111 11111111 11111111
```

Powyżej przedstawiono monochromatyczną bitmapę[36], czyli czarno-białą grafikę złożoną z 144 pikseli (cztery wiersze po 32 piksele) oraz zawartość binarną pliku w formacie BMP, w którym tę grafikę zapisano na dysku. Można zauważyć, że oprócz danych (te 16 bajtów wyróżniono pogrubieniem) plik zawiera tzw. metadane (nagłówek), czyli informację m.in. o tym, że grafika jest monochromatyczna[37].

Odczytywanie przez człowieka ciągu bitów zapisanych z użyciem cyfr 0 i 1, na przykład 101101010011 jest niewygodne i podatne na błędy. W celu uproszczenia tej reprezentacji stosuje się często skrótową notację zwaną notacją szesnastkową (ang. *hexadecimal notation*): każdą czwórkę cyfr 0 lub 1 zastępuje się jedną cyfrą kodu szesnastkowego. Ciągi bitów są w realizacji sprzętowej wielokrotnościami bajtów (oktetów bitów), więc notacja szesnastkowa znakomicie nadaje się.

0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**2**
0	0	1	1	**3**
0	1	0	0	**4**
0	1	0	1	**5**
0	1	1	0	**6**
0	1	1	1	**7**
1	0	0	0	**8**
1	0	0	1	**9**
1	0	1	0	**A**
1	0	1	1	**B**
1	1	0	0	**C**
1	1	0	1	**D**
1	1	1	0	**E**
1	1	1	1	**F**

[36] Czyli bitmapę (dosłownie: mapę bitów) w pierwotnym tego słowa znaczeniu.

[37] Przykłady zadań dla uczniów: 1. Odnaleźć kody binarne poszczególnych wierszy grafiki. 2. Sprawdzić, że wielkość pliku na dysku jest równa liczbie bajtów (tj. 78 bajtów).

Na poprzedniej stronie przedstawiono system szesnastkowy. W lewej kolumnie zapisano wszystkie możliwe ciągi czterobitowe, a w prawej kolumnie odpowiadające im cyfry kodu szesnastkowego. Np. zapis B5 oznaczać będzie 1011 0101.

Aby uzyskać kod szesnastkowy dla ciągu bitów, trzeba najpierw podzielić go na 4-bitowe podciągi, a następnie zapisać każdy z nich za pomocą jego szesnastkowego odpowiednika. Postępując w ten sposób 16-bitowy ciąg 1010 0100 1100 1000 można przedstawić w wygodniejszej (efektywnej) postaci jako A4C8.

```
42 4D 4E 00 00 00 00 00 00 00 3E 00 00 00 28 00
00 00 20 00 00 00 04 00 00 00 01 00 01 00 00 00
00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF 00 FF FF
FF FF CC CC CC CC AA AA AA AA FF FF FF FF
```

Bitmapa w kodzie szesnastkowym (format BMP)

3.1. Kodowanie tekstu

Tekst reprezentuje się w pamięci komputera stosując kod, który kolejne symbole tekstu zastępuje unikatowymi (dla różnych liter, znaków przestankowych itd.) ciągami bitów. Dziś powszechnie używane są kody: ASCII (wym. aski) i Unicode[38].

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	#		$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Podstawowa tablica kodu ASCII

Powyższa tabela zawiera informacje o kodach małych i wielkich liter[39] alfabetu łacińskiego, znaków przestankowych, cyfr, a także pewnych znaków sterujących (takich, jak znak nowego wiersza, znak powrotu karetki i tabulacji[40]). W sumie jest

[38] Praktycznie jedynym wyjątkiem jest kod EBCDIC używany w komputerach *mainframes* koncernu IBM. Kod pochodzi jeszcze z epoki kart dziurkowanych. Więcej w [11, s. 261].

[39] Warto zwrócić uwagę na uporządkowanie kodów cyfr: binarny numer kolumny odpowiada numerycznej wartości danej cyfry, np. 1 znajduje się w kolumnie o numerze $1_{16}=0001_2$.

[40] Pierwsze dwa wiersze (oraz ostatni kod DEL) dotyczą znaków sterujących (ang. *control codes*), które nie mają graficznej postaci (nie widzimy ich na wydruku). Kod ASCII był ko-

ich 128. Wiersze i kolumny są ponumerowane kolejnymi numerami w kodzie szesnastkowym. Np. literę A znajdziemy w tabeli w miejscu przecięcia wiersza o numerze 4 i kolumny o numerze 1, czyli jej kod szesnastkowy to 41, a więc kod binarny to: 0100 0001, który można zapamiętać w jednobajtowej komórce pamięci[41].

Dzięki międzynarodowej współpracy na przełomie XX i XXI w. opracowano kod, zwany Unikod, który pozwala zapisywać teksty z użyciem alfabetów innych niż łaciński (takich jak symbole matematyczne i techniczne, notacja muzyczna itp.).

3.2. Kodowanie wartości liczbowych

Najprostszym sposobem reprezentacji binarnej liczb naturalnych jest tzw. *reprezentacja wagowo-pozycyjna*[42]. Znaczy to, że każda liczba w zapisie ma pewną liczbę pozycji, ponumerowanych od prawej do lewej. Pozycja znajdująca się na prawym końcu ma numer 0, następna w lewo – numer 1, następna – numer 2 itd. Każdej pozycji odpowiada pewna *waga*. W systemie dwójkowym (binarnym) kolejnym pozycjom odpowiadają wagi, które są kolejnymi potęgami liczby 2: 1, 2, 4, 8, 16, itd. Na każdej pozycji należy zapisać jedną z cyfr binarnych (bitów): 0 albo 1.

pozycja	7	6	5	4	3	2	1	0
waga	128	64	32	16	8	4	2	1
bity	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**

Zasada dwójkowego kodowania liczb naturalnych

W powyższej tabeli przedstawiono ideę kodowania na przykładzie liczby, której 8-bitowy kod dwójkowy to 01110011. Jaka to liczba? 64 + 32 + 16 + 2 + 1 = 115.

Ze względu na wartość odpowiadających im wag, bit znajdujący się najdalej z prawej strony (na pozycji nr 0) jest nazywany bitem *najmniej znaczącym*, a bit o największej pozycji – *najbardziej znaczącym* bitem danej liczby.[43]

---

dem telekomunikacyjnym, więc kody sterujące pełniły funkcję jak gdyby wskazówek organizacyjnych w procesie konstruowania i przekazywania wiadomości.

[41] Kod ASCII był pierwotnie zaprojektowany i wprowadzony (w 1963 r.) jako kod 7-bitowy. Wprowadzając 0 na najbardziej znaczącej pozycji odpowiedniego ciągu 7-bitowego otrzymujemy 8-bitowy kod ASCII. Zwiększenie liczby bitów do 8 pozwala na zakodowanie 128 dodatkowych ciągów bitów (z dodatkowym bitem równym 1), co można wykorzystać do reprezentowania symboli takich, jak np. polskie znaki diakrytyczne.

[42] Podrozdział jest wzorowany na wprowadzeniu z książki [11]. Tam można znaleźć również opis metody (algorytmu) ilorazowego przeliczania (konwersji) liczb z systemu dziesiątkowego na dwójkowy. Tutaj ten opis świadomie pomijamy, bo jest dość powszechnie znany.

[43] Czasami mówi się także o bitach „najstarszych" i „najmłodszych". Te określenia są stosowane także w odniesieniu do ciągów bitów, które nie reprezentują liczb.

Używając powyższego kodu w jednym bajcie (ang. *byte*), czyli na ośmiu bitach (piszemy 1B = 8 b), można reprezentować liczby z zakresu od 0 (00000000) do 255 (11111111). Dysponując 2 B, można zakodować liczby naturalne do 65 535.

Należy podkreślić, że w systemie komputerowym, ze względu na budowę pamięci i procesora, stosowane długości danych (a więc również liczb) są wielokrotnościami bajtów, np. w systemie 64-bitowym podstawowymi jednostkami są tzw. słowa mające długość 64 b (8 B).

Kod opisany powyżej jest czasami nazywany kodem naturalnym. Właściwie stanowi on tylko podstawę do wielu technik reprezentacji wartości liczbowych w komputerze. Najczęściej stosowaną w praktyce reprezentacją liczb, pozwalającą na zapis liczb ujemnych, jest oparta na kodzie U2 (uzupełnieniowym do dwóch)[44].

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
0	1	3	2	7	6	4	5	15	14	12	13	8	9	11	10

Konkretny ciąg bitów można interpretować na wiele sposobów. Ukazuje to powyższa tabela: w pierwszym wierszu zapisano kolejne liczby naturalne, w drugim – odpowiadające im kody binarne, w trzeciej znajdziemy wartości dziesiętne kodów z wiersza drugiego interpretowane w kodzie U2. Ostatni wiersz zawiera zaś wartości dziesiętne kodów wiersza drugiego interpretowane w kodzie Gray'a[45].

Warto wspomnieć na koniec o kodzie BCD (ang. *binary-coded decimals*), który w istocie stanowi zbiór ciągów cyfr dziesiętnych kodowanych dwójkowo: każdą liczbę zapisaną dziesiętnie koduje się tak, że każdą jej cyfrę przedstawia się oddzielnie za pomocą czterobitowej liczby, zapisanej zazwyczaj w kodzie naturalnym, np. 654 ma postać 0110 0101 0100. Kod BCD warto stosować szczególnie tam, gdzie dane wprowadza się cyfra po cyfrze z dziesiętnej klawiatury, a wyniki wyświetla się na dziesiętnych wyświetlaczach (np. w domofonach).

3.3. Kodowanie i kompresja obrazów

Zastosowania przetwarzania danych nie ograniczają się do przetwarzania tekstu i danych liczbowych. Przetwarza się np. grafiki, zdjęcia czy filmy.

Techniki reprezentacji obrazu można podzielić na dwie kategorie: techniki wyko-

[44] Układy arytmetyczne współczesnych procesorów są najczęściej oparte na systemie U2.

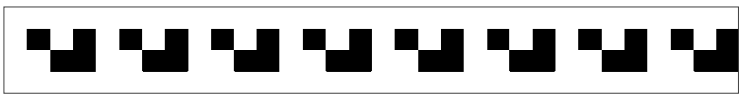
[45] Ciekawym zadaniem dla uczących się może być określenie własności kodu Gray'a. Okazuje się, że to jeden z kodów stosowanych wtedy, gdy wymaga się, aby kody binarne odpowiadające kolejnym liczbom dziesiętnym różniły się wartością tylko jednego bitu. Istnieje wiele takich zastosowań, np. w układach pomiarowych. Por. [1, s. 27-28].

rzystujące mapę bitową (ang. *bit map*) oraz techniki wektorowe[46]. W przypadku pierwszych obraz traktuje się jako zbiór punktów, zwanych pikselami lub pelami (skrót od ang. *picture element* – element obrazu). W najprostszej formie, gdy nie stosuje się żadnej kompresji, obraz monochromatyczny (czarno-biały) reprezentuje się jako długi ciąg bitów (0 lub 1) reprezentujących poszczególne elementy obrazu. Przykład bitmapy został przedstawiony we wprowadzeniu do rozdziału 3.

W przypadku obrazów kolorowych każdemu pikselowi przyporządkowuje się ciąg bitów określających kolor piksela. Urządzenia takie jak skanery czy kamery rejestrują kolor każdego piksela w postaci trzech składowych: czerwonej, zielonej i niebieskiej. Zazwyczaj do zapisu informacji o nasyceniu każdej z tych składowych stosuje się 1 B. Zatem każdy piksel ma reprezentację 24-bitową. Takie rozwiązanie (model kolorów) odpowiada sposobowi wyświetlania obrazu na monitorach.[47]

Dla efektywnego przechowywania danych i ich przesyłania często jest przydatne (z nawet niezbędne) zmniejszanie rozmiaru danych, czyli kompresja danych. Opracowano wiele metod (algorytmów) kompresji danych. Dla każdej z nich można wskazać typy danych, dla których działa ona gorzej i dane, dla których działa najlepiej. Na przykład metoda zwana **kodowaniem długości serii** (ang. *run-length encoding*) daje najlepsze rezultaty, gdy kompresowane dane składają się z długich ciągów tej samej wartości. Metoda polega na zastąpieniu tych ciągów kodem złożonym z pary: wartości, która powtarza się w ciągu oraz liczby wystąpień tej wartości w ciągu. Ta metoda może być świetną metodą kompresji obrazów z dużą liczbą jednolicie zabarwionych bloków (takich jak ikony, znaki firmowe itp.)

Oto przykład zastosowania **kodowania długości serii** dla grafiki czarno-białej (monochromatycznej bitmapy) w wersji stosowanej przez urządzenia typu *fax*: dla każdego wiersza naprzemiennie podaje się liczbę pikseli (peli) białych i czarnych



Faksy łączą kodowanie seria-długość z kodowaniem Huffmana. To metoda tworzenia kodowania zależna od częstości wystąpień. W takich kodach długość ciągu bitów używanego do reprezentacji danego elementu danych jest uzależniona od częstości wystąpień elementu: im częstsze wystąpienie, tym krótszy kod. To oznacza, że różne elementy danych reprezentuje się kodami o różnej długości. Kolejne rzędy peli powyższego obrazu byłyby zakodowane (standard G3) ciągami bitów:

---

[46] Czcionki (fonty) reprezentuje się zwykle wektorowo, czyli jako matematyczne opisy zbioru linii i krzywych tworzących kształt czcionki. Dzięki temu uzyskuje się czcionki skalowalne. Za pomocą technik wektorowych nie jest możliwe uzyskanie obrazów o jakości fotografii.

[47] Tzw. subpiksele można dostrzec na monitorze z użyciem szkła powiększającego.

00011011
000111010**000111**010**…000111**010
011111**0111**11**…0111**11
00011011[48]

Efekt kompresji dla obrazów o większych rozmiarach jest znaczący (nawet 1:7).

3.4. Kody korekcyjne

Są sytuacje, gdy do reprezentacji informacji używamy większej liczby bitów, niż to jest niezbędne. Celem jest zredukowanie prawdopodobieństwa wystąpienia błędu (np. podczas odczytu z jakiegoś nośnika danych lub w czasie transmisji danych z sondy kosmicznej). Kody, w których zastosowana jest jakaś forma redundancji (nadmiarowości) w celu wykrycia błędów (tj. faktu, że np. bit o wartości 0 zastąpił bit o wartości 1) nazywane są kodami detekcyjnymi[49]. Jeśli dane zawierają wystarczająco wiele nadmiarowej informacji, aby wykryć i poprawić takie błędy, to mówimy, że zakodowano je z użyciem kodów korekcyjnych.

Prostym kodowaniem, pozwalającym na wykrycie przekłamania jest *kontrola parzystości*. System ten pozwala wykrywać pojedynczy błąd w danych o dowolnej długości dzięki temu, że do danych źródłowych dodano jeden nadmiarowy bit.

Jako szczególny przykład kodu kontroli parzystości rozważmy 8-bitowy kod ASCII, w którym w zastosowaniach telekomunikacyjnych dodatkowy bit był używany jako *bit parzystości* (ang. *parity bit*): był równy 0 wtedy i tylko wtedy, gdy liczba jedynek w siedmiu pozostałych bitach była parzysta. Oznacza to, że liczba jedynek w 8-bitowym ciągu powinna zawsze być parzysta. Jeśli zakłócenie transmisji spowodowało zamianę 1 na 0 lub na odwrót, to 8-bitowa wiadomość odebrana przez adresata miała nieparzystą liczbę jedynek. Transmisję należało powtórzyć[50].

---

[48] Jak powstał standard? Wybrano zestaw ośmiu reprezentatywnych dokumentów i policzono, jak często dana seria się pojawia. Następnie zbudowano dla tych serii kody Huffmana. Podczas transmisji faksu dołącza się jeszcze kody EOL (000000000001) informujące o nowym wierszu (rzędzie) peli i końcu (powtarza się sześć razy kod EOL). Oczywiście liczba bitów musi być wielokrotnością 8 bitów, więc na końcu dodaje się pewną liczbę 0.

[49] Oczywistą formą redundancji jest przesyłanie informacji więcej niż jeden raz, co pozwala na wykrycie błędów. Jeśli przesłane dwie kopie tej samej informacji różnią się od siebie, to w czasie przesyłania musiał nastąpić jakiś błąd. Zastosowanie prostego kodu korekcyjnego wymaga przesłania informacji trzykrotnie (nadmiarowość 200%). Zakładając, że tylko w jednej informacji wystąpiły zakłócenia, adresat mógłby odtworzyć poprawną wersję, wybierając te dwie identyczne kopie. Na szczęście istnieją kody detekcyjne i kody korekcyjne, funkcjonujące przy znacznie mniejszej redundancji.

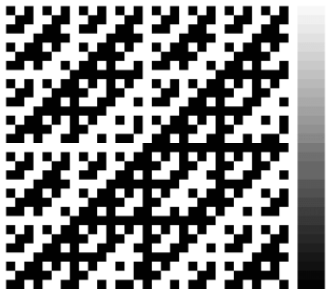
[50] Podobne systemy kontroli parzystości były używane do wykrywania błędów w układach pamięci komputerów. Ograniczeniem tego prostego kodu kontroli jest to, że dobrze wy-

Przykładem kodu, który pozwala na nie tylko na wykrycie pojedynczego przekłamania, ale i automatyczną korektę jest dwuwymiarowy kod kontroli parzystości:

1	0	1	0
0	1	1	0
1	0	0	1
0	1	0	1

W tym przykładzie dziewięć bitów informacji i siedem nadmiarowych bitów parzystości. Bity zawierające informację są ustawione – tak to można sobie wyobrazić – w trzy rzędy. Na każdy wiersz i każdą kolumnę przypada jeden bit parzystości. Możemy więc mówić o kolumnie bitów kontrolnych i wierszu bitów kontrolnych. Błąd w wiadomości polegający na zamianie jednego bitu skutkuje pojawieniem się dwóch naruszeń parzystości: jednego w wierszu, drugiego w kolumnie. Odbiorca wiadomości będzie wtedy wiedział, że bit na przecięciu wadliwego wiersza i wadliwej kolumny jest błędny (i należy go zmienić). Konstrukcja kodu pozwala również na wykrycie i lokalizację błędu przekłamania w przypadku jednego z bitów parzystości przypisanych do wierszy i kolumn (stąd dodatkowy skrajny bit parzystości)[51].

W praktyce w celu zwiększenia niezawodności zapisu i transmisji danych stosuje się bardziej skomplikowane metody kodowania. Pozwalają one na poprawne dekodowanie wartości bitów, nawet jeśli miało miejsce wiele przekłamań[52].



krywa jedynie pojedynczy błąd. Informacja, w której zmienione zostały dwa bity, mieć będzie poprawną parzystość, mimo że dane zawierają błędy.

[51] Takie kody są ciągle używane do ochrony zapisu w pamięciach superkomputerów.

[52] Dla przykładu: w [3] czytamy, że „Podstawowa różnica między pierwotnym formatem CD-DA (Digital Audio) stosowanym w płytach dźwiękowych, a późniejszym formatem CD-ROM zastosowanym do przechowywania danych komputerowych polega na poziomie korekcji błędów. Format CD-DA wykorzystuje techniki korekcji błędów, które redukują możliwość wystąpienia błędu do jednego na dwie płyty. Jest to w zupełności zadowalające w przypadku nagrań dźwiękowych. W przypadku nośników z danymi ryzyko znalezienia błędu na 50% płyt jest nie do zaakceptowania. Dlatego w formacie CD-ROM (i jego pochodnych) zastosowano dodatkowe techniki, które zmniejszają prawdopodobieństwo wystąpienia błędu do jednego na 20 000 płyt."

Przykładem jest kod korekcyjny Reeda-Mullera, zobrazowany powyżej. Kody te, używane w latach 60. XX w. do transmisji z sond kosmicznych zdjęć o odcieniach szarości, pozwalały wykryć (i naprawić!) nawet siedem z 32 przesłanych bitów.

4. Zakończenie

Zadaniem nauczyciela, który ma być przewodnikiem młodego człowieka, stawiającego pierwsze kroki w niełatwym świecie pojęć i metod informatyki, jest wskazanie uczącemu się przykładów, które pozwolą mu na wyrobienie sobie pojęcia o tym, jakie są podstawowe wiadomości i umiejętności wymagane od informatyka. Dotyczy to również zagadnień efektywnego kodowania informacji.

Zaprezentowane powyżej przykłady sposobów reprezentacji informacji (kodowania i kompresji) stanowiły bez wątpienia pierwotnie rozwiązania konkretnych problemów dla konkretnej epoki. Gdy przedstawi się je w odpowiedni sposób (motywując uczniów metodami aktywnymi) nie będą nudnymi *anty*-ciekawostkami *z prehistorii*[53], ale mogą autentycznie zaciekawić i dać uczącym się informatyki dobre wyobrażenie o istocie tego, czym jest rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji. *Historia magistra vitae*[54].
Odważny (choć rozważny) nauczyciel czy nauczycielka nie będą tracić żadnej okazji[55] do wprowadzenia młodego adepta informatyki w świat. Nawet, jeśli sami ten świat odkrywają po raz pierwszy (lub na nowo), przygotowując się do lekcji[56]. Jest tylko jeden warunek wstępny, niezależny od nauczyciela: dostęp do źródeł, materiałów edukacyjnych, z których nauczyciel już sam zrobi dobry użytek.

---

[53] Warto zaznaczyć, że jednym z wymagań szczegółowych podstawy programowej jest: *Uczeń przedstawia główne etapy w historycznym rozwoju informatyki i technologii*.

[54] Warto przypomnieć dydaktyczną zasadę paralelizmu: „Uczenie się jednostki powinno stanowić skrócone powtórzenie uczenia się ludzkości. [Zasada] podpowiada więc czego i w jakiej kolejności uczyć oraz jak uczyć, w jaki sposób." Więcej w [5] s. 127-138.

[55] Do prezentacji różnych zagadnień informatycznych wykorzystają np. lekcje-zastępstwa. Wówczas jest uprzywilejowany czas i miejsce, aby korzystać z metodyki *CS Unplugged*. Więcej na temat nauczania informatyki bez komputera na: http://jasijoasia.edu.pl/

[56] Nauczyciel musi być nieuchronnie przygotowany na to, że uczniowie lub sytuacje będą go stawiać wobec problemów, których rozwiązań z góry nie zna. Na ten temat pisał dr Andrzej Walat, komentując jedną z idei pedagogicznych konstrukcjonizmu Paperta : „Nie wstyd jest przyznać się, że się nie umie. Wstyd przyznać się, że się nie umie nauczyć. (…) Takie zdarzenia przy właściwym podejściu do nich nie tylko nie zakłócają procesu nauki, ale wręcz, bardzo sprzyjają jego skuteczności. W ten sposób uczniowie najskuteczniej uczą się, że trudności są rzeczą naturalną i jak sobie z nimi radzić." Więcej w [17] na s. 24-25.

Warto podjąć ten trud pracy u podstaw. Nic cenniejszego w pracy pedagoga XXI wieku niż widzieć ten błysk zaciekawienia w oku (i wdzięczność) uczącego się.

Literatura

1. Barczak A., Florek J., Sydoruk T., *Elektroniczne techniki cyfrowe*, VIZJA-PRESS&IT, Warszawa 2008.
2. Bauer F. L., Goos G., *Informatyka*, WNT, Warszawa 1977.
3. Brookshear J. G., *Informatyka w ogólnym zarysie*, WNT, Warszawa 2003.
4. Cormen T. H., *Algorytmy bez tajemnic*, Helion, Gliwice 2013.
5. Duda R., *Zasada paralelizmu w dydaktyce*. [w:] *Dydaktyka matematyki* 1/1982.
6. Gleick J., *Informacja*, Znak, Kraków 2012.
7. Hillis W. D., *Wzory na krzemowej płytce*, Wydawnictwo CIS, Warszawa 2000.
8. Kippenhahn R., *Tajemne przekazy*, Prószyński i S-ka, Warszawa 2000.
9. Kodowanie pisma, http://www.wszpwn.com.pl/i10/, kopia w archive.org: http://web.archive.org/web/20051230222319/http://www.wszpwn.com.pl:80/i10/
10. Lunde P., *Tajemnice szyfrów*, Carta Blanca, Warszawa 2009.
11. Mieścicki J., *Wstęp do informatyki*, BTC, Legionowo 2013.
12. http://www.csfieldguide.org.nz/en/teacher/chapters/data-representation.html
13. Sosińska A., *W krainie odkryć i wynalazków*, LSW, Warszawa 1981.
14. Stachura B. (red.), *Informatyka*, WSiP, Warszawa 1976.
15. Sysło M. M. (red.), *Informatyka. Podręcznik dla liceum ogólnokształcącego. Część I*, WSiP, Warszawa 2002.
16. Sysło M.M., Historia rachowania – ludzie, idee, maszyny. Historia mechanicznych kalkulatorów, [w:] *Homo informaticus*, M.M. Sysło (red.), WWSI, Warszawa 2012, str. 275-316.
17. Walat A., *Zarys dydaktyki informatyki*, OEIiZK, Warszawa 2006.