ASCII- *American Standard Code for Information Interchange*

Amerykański Standardowy Kod do Wymiany Informacji

Szyfr Cezara

Komputer przechowuje informacje w formie binarnej, reprezentując je jako sekwencję jedynek i zer. Aby przetłumaczyć informacje na formę wygodną dla ludzkiej percepcji, każda unikalna sekwencja cyfr wyświetlana jest zastępowana przez odpowiedni **znak**. Jednym z systemów korelacji kodów binarnych ze znakami drukowanymi i sterującymi jest kodowanie **ASCII**. Ogólne zrozumienie sposobu kodowania jest niezwykle użyteczne, a dla niektórych kategorii specjalistów (szyfrantów, kryptologów) jest to absolutnie konieczne.

Oryginalne kodowanie (na podstawie kodu telegraficznego) zostało opracowane w 1963 roku (dla dalekopisów), a następnie aktualizowane przez 25 lat. W pierwotnej wersji tablica znaków ASCII zawierała 128 znaków **(7 bitów** 0 - 127), później pojawiła się wersja rozszerzona, w której zapisano pierwszych 128 znaków, a kody z zastosowanym 8-bitem pasują do poprzednio niezaznaczonych znaków (0 -255).

Przyporządkowuje liczbom z zakresu 0−127: litery alfabetu łacińskiego języka angielskiego, cyfry, znaki przestankowe i inne symbole oraz polecenia sterujące. Na przykład litera „a” jest kodowana jako liczba 97, a znak spacji jest kodowany jako 32. Większość współczesnych systemów kodowania znaków jest rozszerzeniem standardu ASCII.

ASCII- *American Standard Code for Information Interchange*

Szyfr Cezara

Zestawy 8-bitowe

Wraz z rozwojem komputerów oraz spadkiem kosztów transmisji danych, zaczęły się pojawiać 8-bitowe zestawy znaków. Pojawiły się standardy z rodziny ISO 8859 oraz Windows-1250, które zapewniały obsługę liter narodowych używając do tego zakresu 128–255, jednocześnie pozostawiając nietknięty obszar wspólny z ASCII. Alfabet polski był wspierany przez kodowanie **ISO 8859-2 (Latin 2)** oraz **Windows-1250**.

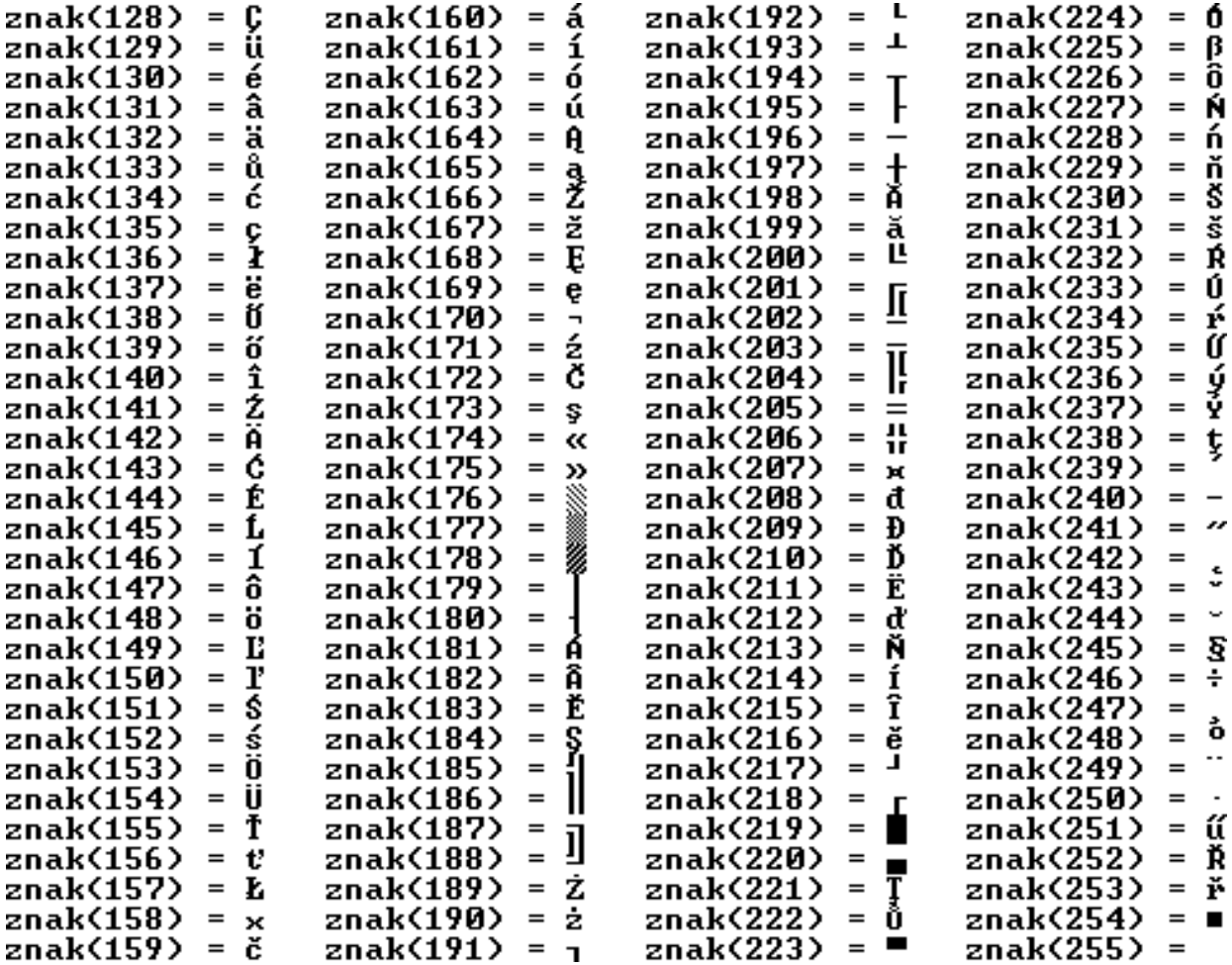
Unicode (UTF-7, UTF-8 – komunikacja sieciowa, Unikod) obsługują znacznie większą liczbę znaków (1 – 4 bajtów), dzięki czemu wszystkie alfabety używane na świecie mogą zostać umieszczone w jednym kodowaniu. Są one wstecznie kompatybilne z ASCII (tekst składający się wyłącznie ze znaków 0–127 ma taką samą reprezentację w obu standardach)[

ASCII TABLE

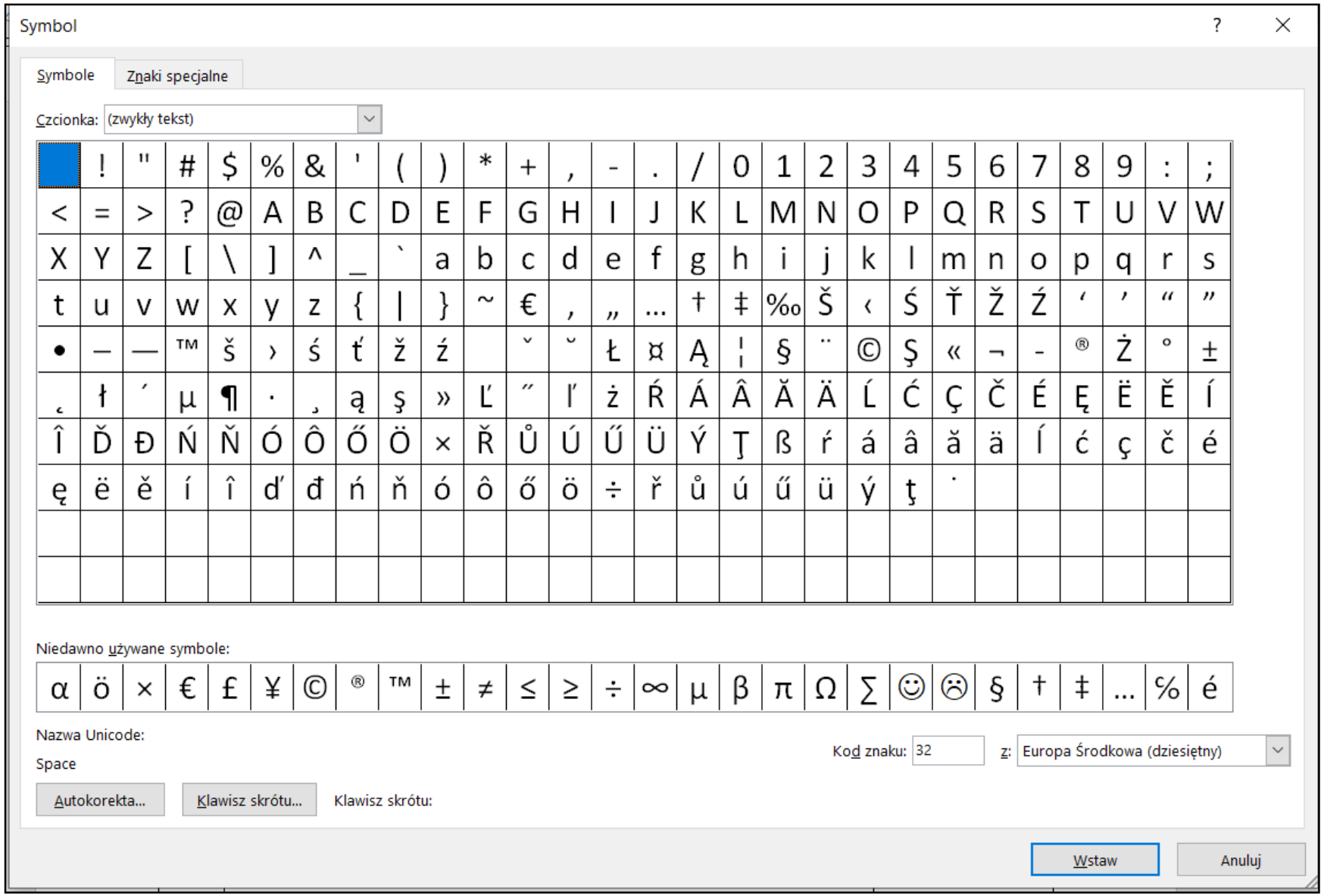
Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

7 – bitowa tabela kodów ASCII.
Znaki od 0 do 127

ASCII rozszerzona do 8 bitów. Znaki od 128 do255



8-bitowy kod pozwala przedstawić 256 różnych wartości i tylko tyle może być zdefiniowane znaków w kodzie ASCII. Pierwsza połówka zbioru kodów – od 0 do 127 – jest zdefiniowana na stałe i raczej nigdy nie jest modyfikowana. Jest to tzw. podstawowy zestaw znaków ASCII. Druga połówka – od 128 do 255 – zawiera **znaki narodowe**, które w różny sposób mogą być przydzielane rozszerzonym kodom ASCII. Z tego właśnie powodu powstały różne strony kodowe. Na przykład konsola znakowa stosuje kodowanie LATIN II. Natomiast system Windows stosuje Windows 1250. Niestety, w obu systemach polskie literki posiadają różne kody. Dlatego wyświetlenie przygotowanego w Windows polskiego tekstu w konsoli znakowej powoduje, iż polskie znaki Windows 1250 zostają źle zinterpretowane w konsoli LATIN II. Problem ten nie występuje w systemach linuksowych.



Tablela znaków Unicode - Europa Środkowa (dziesiętny). Tabela pochodzi z programu Word. Zawiera znaki ASCII oraz znaki Europy Środkowej.

Znaki są zapamiętywane w postaci kodów 16-bitowych. Dzięki temu rozwiązaniu liczba możliwych do przedstawienia znaków rośnie do 65536. Pierwsze 256 kodów jest kompatybilne z kodami ASCII. Kody powyżej 256 tworzą banki znaków, w których znajdują się wszystkie znaki narodowe, arabskie, hebrajskie, matematyczne itp.

www.flynerd.pl

Strona o programowaniu - kurs Pythona

https://www.flynerd.pl/2019/09/kodowanie-znakow-ascii-unicode-utf-co-to-znaczy.html

Wyczerpujący artykuł o kodowaniu

Kodowanie znaków w konsoli

```
#include <iostream>
 using namespace std;
 int main( )
{
 cout << "ĄąĆćĘęŁłŃńÓóŚśŹźŻż\n";
return 0;
}
```

```
#include <iostream>

using namespace std;

int main( )
{
    setlocale ( LC_ALL, "" );

    cout << "Zażółć gęślą jaźń" << endl;
    return 0;
}
```

W języku C++ istnieje funkcja, która ustawia w konsoli znakowej ten sam system kodowania znaków, który stosuje Windows: `setlocale ( LC_ALL, "" );`

Typy znakowe: char i wchar_t

Typy znakowe w C++

ASCII	unsigned char
Unicode	wchar_t

Deklaracje zmiennej w C++

```
char c;
wchar_t wc;
```

Zmienne znakowe mogą również być zadeklarowane jako tablice znaków

```
char c [ 100 ];
wchar_t wc [ 100 ];
```

Program tworzy trzyznakową tablicę i wpisuje do niej wyraz ALO. Następnie literki są wypisywane w kierunku odwrotnym:

```
#include <iostream>

using namespace std;

int main( )
{
  unsigned char s [ 3 ];

  s [ 0 ] = 'A';
  s [ 1 ] = 'L';
  s [ 2 ] = 'O';
  cout << s [ 2 ] << s [ 1 ] << s [ 0 ]
       << endl << endl;
  return 0;
}
```

Typy tablicowe: string, wstring

Oprócz zwykłych **tablic znakowych** (ang. character tables) C++ udostępnia tzw. **łańcuchy znakowe** (ang. character strings). Są to tablice dynamiczne, które mogą przechowywać ciągi znaków o różnych długościach (łańcuchy automatycznie dopasowują się do rozmiaru przechowywanego tekstu – ***tablice znakowe natomiast nie posiadają takich cech***, programista musi o to zadbać sam

Liczbę znaków przechowywanych w łańcuchu tekstowym otrzymamy przy pomocy następujących funkcji:

```
s.length( )
s.size( )
```

```
#include <iostream>
#include <string>

using namespace std;

int main( )
{
  string s;
  s = "Hello World!";
  for( int i = s.length( ) - 1; i >= 0; i-- )
    cout << s [ i ];
  cout << endl << endl;
  return 0;
}
```

Programy demonstrują sposoby deklarowania zmiennej łańcuchowej oraz dostępu do znaków zawartych w łańcuchu

```
#include <iostream>
#include <string>

using namespace std;

int main( )
{
  wstring s;
  s = L"Hello World!";
  for( int i = s.length( ) - 1; i >= 0; i-- )
    cout << ( char )s [ i ];
  cout << endl << endl;
  return 0;
}
```

KOD ZNAKU

Przy przetwarzaniu tekstu często trzeba odczytywać kody znaków zawartych w zmiennej znakowej lub zamieniać kody na odpowiadające im znaki – na przykład w celu umieszczenia ich w tekście.

Język C++ traktuje znaki jak liczby całkowite (unsigned char – bez znaku, char – ze znakiem). Nie ma zatem zwykle potrzeby dokonywać konwersji znakowych. Wyjątek stanowi przesyłanie znaków do strumieni – musimy dokonać konwersji kodu, aby w strumieniu został zapisany znak, a nie jego kod jako liczba całkowita.

Kod dostępu do znaku	(int) znak
Zamiana kodu na znak	char (kod)

```
#include <iostream>
using namespace std;
int main()
{
   setlocale ( LC_ALL, "" );

   char ch = 'M';      // przypisanie zmiennej ch kodu ASCII znaku M
   int i = ch;         // zapis tego samego kodu jako wartości int

   cout << "Kod ASCII znaku " << ch << " to " << i << endl;

   cout << "Dodajemy do kodu znaku jedynkę:" << endl;

   ch = ch + 1;        // zmiana kodu znaku ze zmiennej ch
   i = ch;             // zapis nowego kodu znaku w i
   cout << "Kod ASCII znaku " << ch << " to " << i << endl;

   return 0;
}
```

Rzutowanie

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main( )
{
  char s;
  int k;
cout << "Podaj liczbe znaku" << endl;
cin >> k;
cout << "Liczba  " << k << " to znak " << char(k) << endl;


  return 0;
}
```

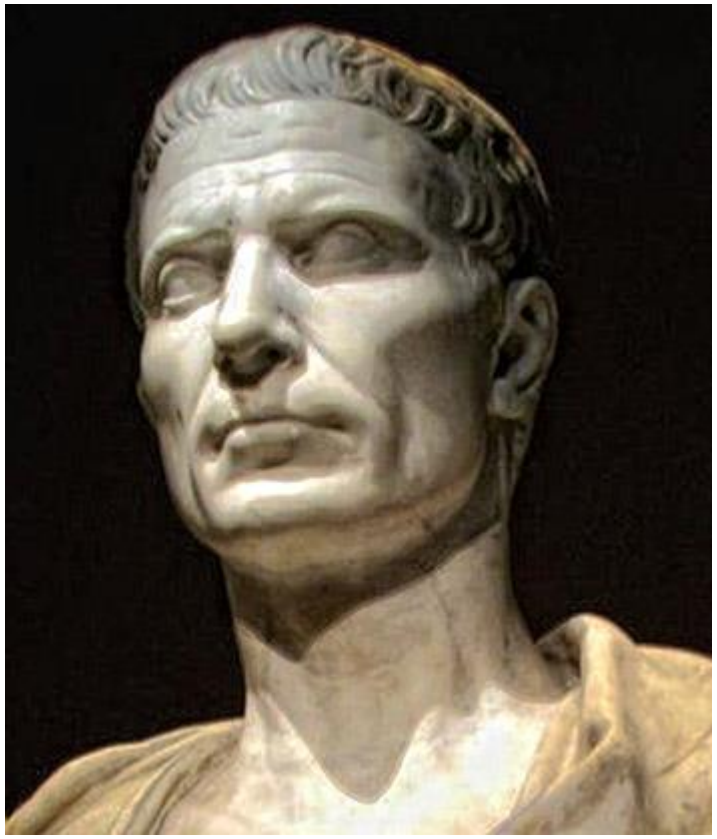
Program odczytuje z kod znaku typu integer ,a następnie rzutuje go na literał typu char zgodnie z kodem ASCII

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main( )
{
  string s;

  getline ( cin, s );
  cout << endl;
  for( unsigned i = 0; i < s.length( ); i++ )
    cout << s [ i ] << ": " << setw ( 3 ) << ( int )s [ i ] << endl;
  cout << endl;
  return 0;
}
```

Program odczytuje z klawiatury łańcuch znaków do zmiennej łańcuchowej, a następnie wypisuje kolejne literki wraz z ich kodami ASCII

Szyfr Cezara – szyfr podstawieniowy

Szyfrowanie tekstów (ang. text encryption) ma na celu ukrycie ważnych informacji przed dostępem do nich osób niepowołanych. Historia kodów szyfrujących sięga czasów starożytnych. Już tysiące lat temu egipscy kapłani stosowali specjalny system hieroglifów do szyfrowania różnych tajnych wiadomości. Szyfr Cezara (ang. Ceasar's Code lub Ceasar's Cipher) jest bardzo prostym szyfrem podstawieniowym (ang. substitution cipher). Szyfry podstawieniowe polegają na zastępowaniu znaków tekstu jawnego (ang. plaintext) innymi znakami przez co zawarta w tekście informacja staje się nieczytelna dla osób niewtajemniczonych. Współcześnie szyfrowanie stanowi jedną z najważniejszych dziedzin informatyki – to dzięki niej stał się możliwy handel w Internecie, funkcjonują banki ze zdalnym dostępem do kont, powstał podpis elektroniczny oraz bezpieczne łącza transmisji danych.



Szyfr Cezara

Szyfr Cezara został nazwany na cześć rzymskiego imperatora Juliusza Cezara, który stosował ten sposób szyfrowania do przekazywania informacji o znaczeniu wojskowym. Szyfr polega na zastępowaniu liter alfabetu A...Z literami leżącymi o trzy pozycje dalej w alfabecie:

Tekst jawny	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	**X**	**Y**	**Z**
Szyfr Cezara	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	**A**	**B**	**C**

Ostatnie trzy znaki X, Y i Z nie posiadają następników w alfabecie przesuniętych o trzy pozycje. Dlatego umawiamy się, iż alfabet "zawija się" i za literką Z następuje znów litera A. Teraz bez problemu znajdziemy następniki X → A, Y → B i Z → C.

Przykład:
Zaszyfrować zdanie: NIEPRZYJACIEL JEST BARDZO BLISKO.
Poszczególne literki tekstu jawnego zastępujemy literkami szyfru Cezara zgodnie z powyższą tabelką kodu. Spacje oraz inne znaki nie będące literami pozostawiamy bez zmian:

Szyfr Cezara

NIEPRZYJACIEL JEST BARDZO BLISKO

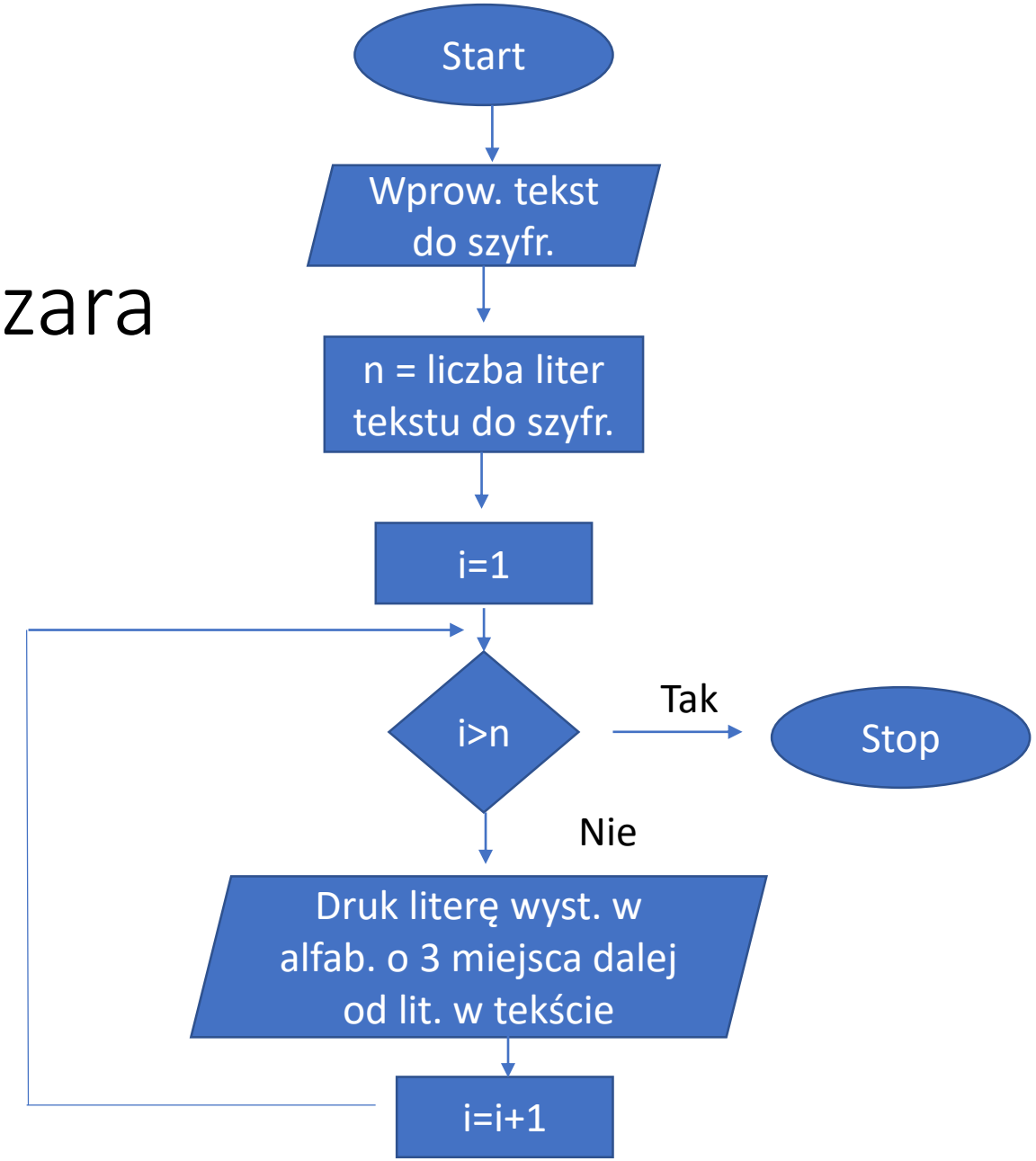
QLHSUCBMDFLHO MHVW EDUGCR EOLVNR

Deszyfrowanie tekstu zaszyfrowanego kodem Cezara polega na wykonywaniu operacji odwrotnych. Każdą literę kodu zamieniamy na literę leżącą o trzy pozycje wcześniej w alfabecie.

Szyfr Cezara	**A**	**B**	**C**	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Tekst jawny	**X**	**Y**	**Z**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Podobnie jak poprzednio trzy pierwsze znaki szyfru Cezara nie posiadają bezpośrednich odpowiedników liter leżących o trzy pozycje wcześniej, ponieważ alfabet rozpoczyna się dopiera od pozycji literki D. Rozwiązaniem jest ponowne "zawinięcie" alfabetu tak, aby przed literą A znalazły się trzy ostatnie literki X, Y i Z.
Do wyznaczania kodu literek przy szyfrowaniu i deszyfrowaniu posłużymy się operacjami modulo. Operacja modulo jest resztą z dzielenia danej liczby przez moduł. Wynik jest zawsze mniejszy od modułu. U nas moduł będzie równy 26, ponieważ tyle mamy liter alfabetu.

Schemat blokowy Cezara



Szyfr Cezara

Jeśli c jest kodem ASCII dużej litery alfabetu (rozważamy tylko teksty zbudowane z dużych liter), to szyfrowanie kodem Cezara polega na wykonaniu następującej operacji arytmetycznej:

$c = 65 + (c - 62) \textbf{ mod } 26$

Deszyfrowanie polega na zamianie kodu wg wzoru:

$c = 65 + (c - 42) \textbf{ mod } 26$

Szyfr Cezara – szyfrowanie

65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	Kod ASCII
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	Znak
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Indeks tablicy

Klucz 3 (65-3 =62)

C = 65 + (C - 62) **mod** 26
C = 65 + (67 -62) mod 26
C = 65 + (5 mod 26)
C = 65 + 5 = 70
C = 70
C = F

Kod szyfruje o 3 miejsca w prawo znak C. Trzy miejsca pomniejszając liczbę kodu A o trzy: 65-3=62. Następnie od kodu C odejmujemy 62: 67-62=5. Liczbę 5 dzielimy przez moduł 26 (liczba znaków w alfabecie) i otrzymujemy 5. Potem do kodu znaku A , czyli 65 dodajemy 5 = 70 czyli kod znaku F.
Znak **C** został zaszyfrowany na znak **F**

s [i] = char (65 + (s [i] - 62) % 26); dla znaku A:
s[0] = char (65 + (65 -62) % 26 = 65+(3%26)= 65+3=68
s[0] = char 68 = D

s [i] = char (65 + (s [i] - 62) % 26); dla znaku B:
s[1] = char (65 + (66 -62) % 26 = 65+(4%26)= 65+4=69
s[1] = char 69 = E

s [i] = char (65 + (s [i] - 62) % 26); dla znaku C:
s[2] = char (65 + (67 -62) % 26 = 65+(5%26)=65+5=70
s[2] = char 70 = F

Klucz szyfrowania w tych działaniach wynosi **3**.
Litera szyfrowana – (65-**3**=62) mod 26

Szyfr Cezara – deszyfracja

65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	Kod ASCII
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	Znak
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Indeks tablicy

Klucz 3 (26-3=23), 65-23=42

C = 65 + (C - 42) **mod** 26
C = 65 + (67 – 42) mod 26
C = 65 + (25 mod 26)
C = 65 + 25= 90
C = Z

Znak Z jest deszyfrowany , „przywracany” do miejsca przed szyfrowaniem. Kod deszyfruje (przesuwa) znak Z o 3 miejsca w lewo, ale „liczbowo” o 23 miejsca w prawo. Liczba 42 jest wynikiem różnicy liczb 65 i 23 = 42. Liczba 23 , to o 3 miejsca pomniejszona liczba alfabetu, czyli klucz deszyfracji wynosi 3. Gdyby wynosił np. 4 (przesunięcie o 4) to trzeba od 26 – 4 = 22 i od 65 odjąć 22 =43 i od 67 – 43 mod 26 . Wtedy C = 65 + (67 – 43) mod 26 = 65+ 24 mod 26= 65+24=89 =Y

s [i] = char (65 + (s [i] - 42) % 26); dla znaku A:
s[0] = char (65 + (65 -42) % 26 = 65+(23%26)= 65+23=88
s[0] = char 88 = X

s [i] = char (65 + (s [i] - 42) % 26); dla znaku B:
s[1] = char (65 + (66 -42) % 26 = 66+(24%26)= 66+24=89
s[1] = char 89= Y

s [i] = char (65 + (s [i] - 42) % 26); dla znaku C:
s[2] = char (65 + (67 -42) % 26 = 65+(25%26)= 65+25=90
s[1] = char 90 = Z

```
#include <iostream>
#include <string>

using namespace std;

int main( )
{
  string s;
  int i;

// odczytujemy wiersz znaków

  getline ( cin, s );

// zamieniamy małe litery na duże
// i kodujemy szyfrem cezara

  for( i = 0; i < s.length( ); i++ )
  {
    s [ i ] = toupper ( s [ i ] );
    if( ( s [ i ] >= 'A' ) && ( s [ i ] <= 'Z' ) )
    s [ i ] = char ( 65 + ( s [ i ] - 62 ) % 26 );
  }

// wypisujemy zaszyfrowany tekst

  cout << s << endl << endl;
  return 0;
}
```

Program szyfrujący

```
#include <iostream>
#include <string>

using namespace std;

int main( )
{
  string s;
  int i;

// odczytujemy wiersz znaków

  getline ( cin, s );

// zamieniamy małe litery na duże
// i kodujemy szyfrem cezara

  for( i = 0; i < s.length( ); i++ )
  {
    s [ i ] = toupper ( s [ i ] );
    if( ( s [ i ] >= 'A' ) && ( s [ i ] <= 'Z' ) )
    s [ i ] = char ( 65 + ( s [ i ] - 42 ) % 26 );
  }

// wypisujemy zaszyfrowany tekst

  cout << s << endl << endl;
  return 0;
}
```

Program deszyfrujący

Linki do OKI. Zadanie Szyfr Cezara-szkopuł

Kompletna strona

https://oki.org.pl/szyfr-cezara/

Film na YT

https://oki.org.pl/szyfr-cezara/

```
#include <iostream>
using namespace std;

int main() {
 int typ_dzialania, klucz, i, liczba_znakow;
 string wiadomosc;
 char znak;

 cin >> typ_dzialania; //1 szyfruj (dodawaj klucz)    2 deszyfruj (odejmuj klucz)
 cin >> klucz;
 cin >> wiadomosc;

 liczba_znakow = 'Z' - 'A' + 1;
 for (i=0; i<wiadomosc.length(); ++i) {
                                znak = wiadomosc[i];
                                if ( (znak >= 'A') && (znak <= 'Z') ) {
                                  if (typ_dzialania==1) {
       if ((int)znak+klucz <= (int)'Z')
         znak = znak + klucz;
       else
         znak = znak + klucz - liczba_znakow;
                                } else {
       if ((int)znak-klucz >= (int)'A')
         znak = znak - klucz;
       else
         znak = znak - klucz + liczba_znakow;
                                }
                               }
   if ( (znak>='a') && (znak<='z') ) {
     if (typ_dzialania==1) {
       if ((int)znak+klucz <= (int)'z')
         znak = znak + klucz;
       else
         znak = znak + klucz - liczba_znakow;
                                } else {
       if ((int)znak-klucz >= (int)'a')
         znak = znak - klucz;
       else
         znak = znak - klucz + liczba_znakow;
                                }
                               }
   wiadomosc[i] = znak;
 }

 cout << wiadomosc << "\n";

 return 0;
}
```