

Reprezentacja symboli w komputerze.

Znaki alfabetu i łańcuchy znakowe.

- 7 bitów, liczby z zakresu 0-127
- litery (alfabet angielski) [a-zA-Z]
- cyfry [0-9],
- znaki przestankowe, np. , . ; '
- inne symbole drukowalne, np. * ^ \$
- białe znaki: spacja, tabulacja, ...
- polecenia sterujące: znak nowej linii \n, nowej strony, koniec tekstu, koniec transmisji,...
- 8-bitowe rozszerzenia ASCII: cp1250, latin2, ...

n	znak	n	znak	n	znak	n	znak	n	znak	n	znak
0	NUL '\0'	22	SYN	44	,	66	B	88	X	110	n
1	SOH	23	ETB	45	-	67	C	89	Y	111	o
2	STX	24	CAN	46	.	68	D	90	Z	112	p
3	ETX	25	EM	47	/	69	E	91	[	113	q
4	EOT	26	SUB	48	0	70	F	92	\	114	r
5	ENQ	27	ESC	49	1	71	G	93	]	115	s
6	ACK	28	FS	50	2	72	H	94	^	116	t
7	BEL '\a'	29	GS	51	3	73	I	95	_	117	u
8	BS '\b'	30	RS	52	4	74	J	96	'	118	v
9	HT '\t'	31	US	53	5	75	K	97	a	119	w
10	LF '\n'	32	SPACE	54	6	76	L	98	b	120	x
11	VT '\v'	33	!	55	7	77	M	99	c	121	y
12	FF '\f'	34	"	56	8	78	N	100	d	122	z
13	CR '\r'	35	#	57	9	79	O	101	e	123	{
14	SO	36	\$	58	:	80	P	102	f	124	
15	SI	37	%	59	;	81	Q	103	g	125	}
16	DLE	38	&	60	<	82	R	104	h	126	~
17	DC1	39	'	61	=	83	S	105	i	127	DEL
18	DC2	40	(	62	>	84	T	106	j		
19	DC3	41	)	63	?	85	U	107	k		
20	DC4	42	*	64	@	86	V	108	l		
21	NAK	43	+	65	A	87	W	109	m		

- typ char, 1 bajt, liczba całkowita $[-128, 127]$
- **stała znakowa** - znak zapisany w apostrofach, np.:
 - 'A' to liczba 65,
 - 'A'+1 to liczba 66 (kod litery 'B'),
 - nowy wiersz '\n' to liczba 10,
 - tabulator '\t' to liczba 9,
 - znak '\0' ma wartość 0
 - powrót karetki \r (w MS Windows koniec linii to \r\t)
 - znaki o specjalnym zapisie:
 - ukośnik w lewo '\\',
 - apostrof '\'',
 - cudzysłów '\"',
 - znak zapytania '\?'
- Uwaga: "A" nie jest pojedynczym znakiem lecz tablicą znaków!

```
1  #include<stdio.h>
2
3  int main()
4  {
5      char a;
6
7      printf("Podaj znak: ");
8      scanf("%c",&a);
9
10     printf("znak %c, dec %d, oct %o, hex %x\n",a,a,a,a);
11     return 0;
12 }
```

 char.c

WYKRYWANIE MAŁEJ LITERY

```
int islower(int x)
{
    if(x >= 'a' && x <= 'z' ) return 1;
    return 0;
}
```

ZAMIANA MAŁEJ LITERY NA DUŻĄ

```
int toupper(int x)
{
    if( islower(x) ) x = x + 'a' - 'A';
    return x;
}
```

Plik nagłówkowy: `ctype.h`

KLASYFIKACJA ZNAKÓW

- `isalnum()` - litera alfabetu lub cyfra [A-Za-z0-9]
- `isalpha()` - litera alfabetu [A-Za-z]
- `islower()`, `isupper` - mała / duża litera alfabetu
- `isdigit()` - cyfra
- `isgraph()` - znaki drukowalne (ze spacją)
- `isspace()` - znaki białe
- `isprint()` - znaki drukowalne (bez spacji)
- `ispunct()` - znaki przestankowe (drukowalne bez liter i cyfr)

MANIPULACJE NA ZNAKACH

- `tolower` , `toupper` - zamiana wielkości liter alfabetu

PRZYKŁADY W C: ZAMIANA WIELKOŚCI ZNAKÓW

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int main()
5  {
6      int a;
7
8      do
9      {
10         a=getchar();
11         putchar(toupper(a));
12     }while( a != EOF );
13
14     return 0;
15 }
```

 toupper2.c

- funkcja `getchar()`
zwraca pojedynczy znak
ze standardowego wejścia
lub EOF
- funkcja `putchar()`
wypisuje znak
- EOF - koniec pliku
(*end of file*)
- `toupper2 < plik.txt`
- EOF w terminalu:
CTRL+Z (Windows)
CTRL+D (Unix/Linux)

- **Łańcuch** (*string*) - skończony ciąg symboli alfabetu
- W języku C brak typu `string`, występuje w C++
- Łańcuch to tablica zawierająca ciąg znaków zakończony znakiem `\0` (wartość zero)

A	l	a		m	a		k	o	t	a	\0
---	---	---	--	---	---	--	---	---	---	---	----

- Dostęp do znaku jak w tablicach: `t[i]`
- Stała napisowa (literał znakowy): `"Ała ma kota"`
- stała znakowa `'A'` to liczba 65
- stała napisowa `"A"` to tablica

A	\0
---	----

```
#include<stdio.h>

int main()
{
    char *s1="nieciekawy fragment tekstu.";
    char s2[]="Ala ma kota";

    s1[0]='N';
    s2[0]='E';

    printf(s1);
    printf(" %s\n",s2);

    printf("\nBardzo %s\n", s1+3);

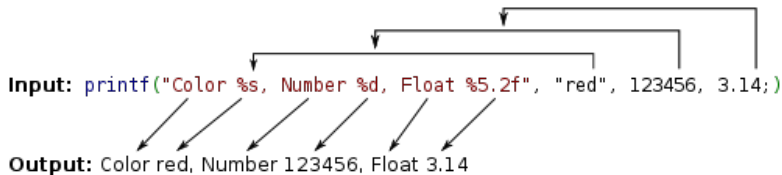
    return 0;
}
```

s1 to stały napis

Źle!

```
int printf(char *s, ...);
```

- wypisuje napis zgodnie z zadany formatem



SPECYFIKACJA FORMATU

`%[flagi][szerokość][.precyzja][długość]specyfikator`

<http://wikipedia.org>

%[flagi] [szerokość] [.precyzja] [długość]specyfikator

specyfikator	znaczenie	przykład
d lub i	liczba całkowita ze znakiem	-123
u	liczba całkowita bez znaku	123
o	liczba całkowita bez znaku ósemkowa	123
x lub X	liczba całkowita bez znaku szesnastkowo	fa1 FA1
f lub F	zmiennopozycyjna w zapisie dziesiętnym	123.45678
e lub E	notacja naukowa	1.2345678e+2
g lub G	krótszy zapis %f lub %e	123.4657
c	pojedynczy znak (ASCII)	a
s	łańcuch znakowy	Ala ma kota
p	adres (wskaźnik), szesnastkowo	ff01ffab
%	wypisuje znak %	%

%[flagi] [szerokość] [.precyzja] [długość]specyfikator

flaga	znaczenie	przykład
-	wyrównanie do lewej (względem podanej szerokości)	123
+	liczby dodatnie poprzedzone znakiem +	+123
<i>spacja</i>	wstawia spację zamiast znaku +	123
0	wypełnia podaną szerokość znakiem 0	000123
#	liczby ósemkowe i szesnastkowe poprzedza 0, 0x, 0X	0123 0xfa1

szerokość	znaczenie
<i>liczba</i>	określa minimalną ilość znaków wypisanych (szerokość pola), brakujące miejsca są dopełniane spacjami. Jeżeli szerokość pola jest za mała to wynik nie jest obcinany.
*	szerokość nie jest dana w formacie lecz poprzez argument funkcji <code>printf</code>

`%[flagi] [szerokość] [.precyzja] [długość] specyfikator`

precyzja	znaczenie
<code>.liczba</code>	ilość cyfr wypisywanych po przecinku
<code>.*</code>	precyzja liczby zmiennopozycyjnej nie jest podawana w formacie lecz poprzez argument funkcji <code>printf</code> . W przypadku łańcuchów znakowych oznacza maksymalną liczbę wypisanych znaków (łańcuch jest obcinany)

długość	znaczenie
<code>brak</code>	liczby <code>int</code> , <code>double</code> , <code>float</code>
<code>l</code>	liczby <code>long int</code>
<code>ll</code>	liczby <code>long long int</code>
<code>L</code>	liczby <code>long double</code>

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf ("Znaki                : %c %c \n", 'A', 65);
6      printf ("Liczby calkowite     : %d \n", 123);
7      printf ("   ze znakiem             : %+d \n", 123);
8      printf ("   szesnastkowo              : %x %#x \n", 123, 123);
9      printf ("   dopelnienie spacjami    : %20d \n", 123);
10     printf ("   dopelnienie zerami      : %020d \n", 123);
11     printf ("Zmienopozycyjne           : %f \n", 3.1416);
12     printf ("   notacja naukowa         : %e \n", 3.1416);
13     printf ("   precyzja                 : %.3f \n", 3.1416);
14     printf ("   dopelnienie              : %20.3f \n", 3.1416);
15     printf ("Szerokosc pola   *        : %*d \n", 20, 123);
16     printf ("Precyzja   .*            : %20.*f \n", 3, 3.1416);
17     printf ("Napis                     : %s \n", "Ala ma kota");
18     printf ("   dopelnienie             : %20s \n", "Ala ma kota");
19     printf ("   obciecie                 : %20.5s \n", "Ala ma kota");
20
21     return 0;
22 }

```

 printf.c

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf ("Znaki                : %c %c \n", 'A', 65);
6      printf ("Liczby calkowite      : %d \n", 123);
7      printf ("   ze znakiem                : %+d \n", 123);
8      printf ("   szesnastkowo              : %x %#x \n", 123, 123);
9      printf ("   dopelnienie spacjami      : %20d \n", 123);
10     printf ("   dopelnienie zerami       : %020d \n", 123);
11     printf ("Zmienopozycyjne            : %f \n", 3.1416);
12     printf ("   notacja naukowa          : %e \n", 3.1416);
13     printf ("   precyzja                  : %.14f \n", 3.1416);
14     printf ("   dopelnienie               : %.14f \n", 3.1416);
15     printf ("Szerokosc pola   *         : %10d \n", 123);
16     printf ("Precyzja   .*           : %.14f \n", 3.1416);
17     printf ("Napis                  : %s \n", "abc");
18     printf ("   dopelnienie            : %10s \n", "abc");
19     printf ("   obciecie                : %.5s \n", "abc");
20
21     return 0;
22 }

```

Znaki	: A A
Liczby calkowite	: 123
ze znakiem	: +123
szesnastkowo	: 7b 0x7b
dopelnienie spacjami	: 123
dopelnienie zerami	: 00000000000000000000123
Zmienopozycyjne	: 3.141600
notacja naukowa	: 3.141600e+00
precyzja	: 3.142
...	

printf.c

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf ("Znaki
6      printf ("Liczby calkowite
7      printf ("   ze znakiem
8      printf ("   szesnastkowo
9      printf ("   dopelnienie spacjami
10     printf ("   dopelnienie zerami
11     printf ("Zmienopozycyjne
12     printf ("   notacja naukowa
13     printf ("   precyzja
14     printf ("   dopelnienie
15     printf ("Szerokosc pola *
16     printf ("Precyzja .*
17     printf ("Napis
18     printf ("   dopelnienie
19     printf ("   obciecie
20
21     return 0;
22 }

```

Zmienopozycyjne	:	3.141600
notacja naukowa	:	3.141600e+00
precyzja	:	3.142
dopelnienie	:	3.142
Szerokosc pola *	:	123
Precyzja .*	:	3.142
Napis	:	Ala ma kota
dopelnienie	:	Ala ma kota
obciecie	:	Ala m

```

: %*d \n", 123);
: %x %#x \n", 123, 123);
: %20d \n", 123);
: %020d \n", 123);
: %f \n", 3.1416);
: %e \n", 3.1416);
: %.3f \n", 3.1416);
: %20.3f \n", 3.1416);
: %*d \n", 20, 123);
: %20.*f \n", 3, 3.1416);
: %s \n", "Ala ma kota");
: %20s \n", "Ala ma kota");
: %20.5s \n", "Ala ma kota");

```

 printf.c

PODSTAWOWE OPERACJE NA ŁAŃCUCHACH

- plik nagłówkowy `string.h`
- długość łańcucha
`int strlen(const char *s);`
- łączenie dwóch łańcuchów
`char *strcat(char *dest, const char *src);`
- porównywanie łańcuchów
`int strcmp(const char *s1, const char *s2);`
- kopiowanie łańcuchów
`char *strcpy(char *dest, const char *src);`
- wyszukiwanie wzorca
`char *strstr(const char *napis, const char *wzor);`

WYSZUKIWANIE WZORCA

Problem: wyszukiwanie podciągu (*pattern matching*).

W ciągu $\mathcal{T}$ znajdź wystąpienie wzorca $\mathcal{W}$.

Tekst $\mathcal{T}$ = "programowanie"

p	r	o	g	r	a	m	o	w	a	n	i	e	\0
---	---	---	---	---	---	---	---	---	---	---	---	---	----

Wzorzec $\mathcal{W}$ = "gra"

g	r	a	\0
---	---	---	----

Algorytm 1 Naiwne wyszukiwanie wzorca

Dane wejściowe: łańcuch znaków $\mathcal{T}$, wzorzec $\mathcal{W}$

Wynik: pozycja tekstu $\mathcal{W}$ w $\mathcal{T}$ lub wartość -1, gdy brak

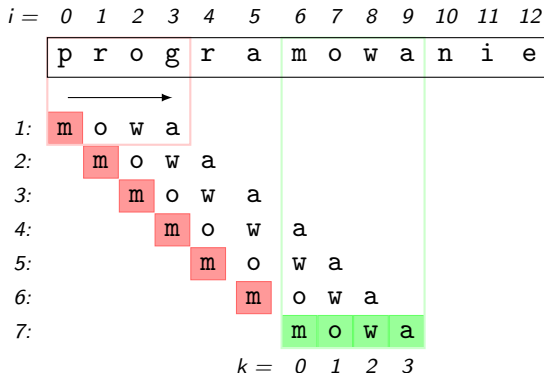
- 1: **dla każdego** $i = 0, 1, 2, \dots, |\mathcal{T}| - |\mathcal{W}|$ **wykonuj**
 - 2: $k \leftarrow 0$
 - 3: **dopóki** $\mathcal{T}[i + k] = \mathcal{W}[k]$ **i** $k < |\mathcal{W}|$ **wykonuj**
 - 4: $k \leftarrow k + 1$
 - 5: **jeżeli** $k = |\mathcal{W}|$ **wykonaj**
 - 6: **zwróć** i
 - 7: **zwróć** -1
-

$|\mathcal{W}|$ oznacza długość łańcucha $\mathcal{W}$

ALGORYTM NAIWNY

Tekst $\mathcal{T}$ = "programowanie"

Wzorzec $\mathcal{W}$ = "mowa"



```
1 int strindex(char t[], char w[])
2 {
3     int i, k;
4
5     i=0;
6     while(t[i] != '\0')
7     {
8         k=0;
9         while(t[i+k]==w[k] && w[k] != '\0')
10             k = k + 1;
11         if(w[k]=='\0') return i;
12         i = i + 1;
13     }
14     return -1;
15 }
```

- Typ znakowy jest liczbą całkowitą
- Łańcuch to tablica znaków zakończona znakiem `'\0'`
- Stała znakowa w apostrofach `'A'`
- Stała napisowa w cudzysłowach `"A"` jest tablicą
- Porównywanie łańcuchów: `t == "napis"` źle, trzeba znak po znaku (funkcja `strcmp()`)
- Kopiowanie napisów: `t = "napis"` źle, kopiowanie tablic (funkcja `strcpy()`)

-  Linux Programmer's Manual
man 7 ascii unicode codepages iso_8859-2
<http://www.unix.com/man-page/>
-  Wikipedia:  Kodowanie polskich znaków diakrycznych
-  Jerzy Wałaszek, „*Algorytmy. Struktury danych.*”,
 Łańcuchy znakowe.
-  R.S. Boyer, J. Strother Moore, *A Fast String Searching Algorithm* Communications of the Association for Computing Machinery, 20(10), 1977, pp. 762-772.
www.cs.utexas.edu/~moore/publications/fstrpos.pdf