

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, ...

USASCII code chart

b7 b6 b5					0 0 0 0 1 0 1 1 0 1 1 0 1 1							
b4 b3 b2 b1					Column Row							
0					1							
1					2							
2					3							
3					4							
4					5							
5					6							
6					7							
7					8							
8					9							
9					10							
10					11							
11					12							
12					13							
13					14							
14					15							
15					16							
16					17							
17					18							
18					19							
19					20							
20					21							
21					22							
22					23							
23					24							
24					25							
25					26							
26					27							
27					28							
28					29							
29					30							
30					31							
31					32							
32					33							
33					34							
34					35							
35					36							
36					37							
37					38							
38					39							
39					40							
40					41							
41					42							
42					43							
43					44							
44					45							
45					46							
46					47							
47					48							
48					49							
49					50							
50					51							
51					52							
52					53							
53					54							
54					55							
55					56							
56					57							
57					58							
58					59							
59					60							
60					61							
61					62							
62					63							
63					64							
64					65							
65					66							
66					67							
67					68							
68					69							
69					70							
70					71							
71					72							
72					73							
73					74							
74					75							
75					76							
76					77							
77					78							
78					79							
79					80							
80					81							
81					82							
82					83							
83					84							
84					85							
85					86							
86					87							
87					88							
88					89							
89					90							
90					91							
91					92							
92					93							
93					94							
94					95							
95					96							
96					97							
97					98							
98					99							
99					100							
100					101							
101					102							
102					103							
103					104							
104					105							
105					106							
106					107							
107					108							
108					109							
109					110							
110					111							
111					112							
112					113							
113					114							
114					115							
115					116							
116					117							
117					118							
118					119							
119					120							
120					121							
121					122							
122					123							
123					124							
124					125							
125					126							
126					127							
127					128							
128					129							
129					130							
130					131							
131					132							
132					133							
133					134							
134					135							
135					136							
136					137							
137					138							
138					139							
139					140							
140					141							
141					142							
142					143							
143					144							
144					145							
145					146							
146					147							
147					148							
148					149							
149					150							
150					151							
151					152							
152					153							
153					154							
154					155							
155					156							
156					157							
157					158							
158					159							
159					160							
160					161							
161					162							
162					163							
163					164							
164					165							
165					166							
166					167							
167					168							
168					169							
169					170							
170					171							
171					172							
172					173							
173					174							
174					175							
175					176							
176					177							
177					178							
178					179							
179					180							
180					181							
181					182							
182					183							
183					184							
184					185							
185					186							
186					187							
187					188							
188					189							
189					190							
190					191							
191					192							
192					193							
193					194							
194					195							
195					196							
196					197							
197					198							
198					199							
199					200							
200					201							
201					202							
202					203							
203					204							
204					205							
205					206							
206					207							
207					208							
208					209							
209					210							
210					211							
211					212							
212					213							
213					214							
214					215							
215					216							
216					217							
217					218							
218					219							
219					220							
220					221							
221					222							
222					223							
223					224							
224					225							
225					226							
226					227							
227					228							
228					229							
229					230							
230					231							
231					232							
232					233							
233					234							
234					235							
235					236							
236					237							
237					238							
238					239							
239					240							
240					241							
241					242							
242					243							
243					244							
244					245							
245					246							
246					247							
247					248							
248					249							
249					250							
250					251							
251					252							
252					253							
253					254							
254					255							
255					256							
256					257							
257					258							
258					259							
259					260							
260					261							
261					262							
262					263							
263					264							
264					265							
265					266							
266					267							
267					268							
268					269							
269					270							
270					271							
271					272							
272					273							
273					274							
274					275							
275					276							
276					277							
277					278							
278					279							
279					280							
280					281							
281					282							
282					283							
283					284							
284					285							
285					286							
286					287							
287					288							
288					289							
289					290							
290					291							
291					292							
292					293							
293					294							
294					295							
295					296							
296					297							
297					298							
298					299							
299					300							
300					301							
301					302							
302					303							
303					304							
304					305							
305					306							
306					307							
307					308							
308					309							
309					310							
310					311							
311					312							
312					313							
313					314							
314					315							
315					316							
316					317							
317					318							
318					319							
319					320							
320					321							
321					322							
322					323							
323					324							
324					325							
325					326							
326					327							
327					328							
328					329							
329					330							
330					331							
331					332							
332					333							
333					334							
334					335							
335					336							
336					337							
337					338							
338					339							
339					340							
340					341							
341					342							
342					343							
343					344							
344					345							
345					346							
346					347							
347					348							
348					349							
349					350							
350					351							
351					352							
352					353							
353					354							
354					355							
355					356							
356					357							
357					358							
358					359							
359					360							
360					361							
361					362							
362					363							
363					364							
364					365							
365					366							
366					367							
367					368							
368					369							
369					370							
370					371							
371					372							
372					373							
373					374							
374					375							
375					376							
376					377							
377					378							
378					379							
379					380							
380					381							
381					382							
382					383							
383					384							
384					385							
385					386							
386					387							
387					388							
388					389							
389					390							
390					391							
391					392							
392					393							
393					394							
394					395							
395					396							
396					397							
397					398							
398					399							
399					400							
400					401							
401					402							
402					403							
403					404							
404					405							
405					406							
406					407							
407					408							
408					409							
409					410							
410					411							
411					412							
412					413							
413					414							
414					415							
415					416							
416					417							
417					418							
418					419							
419					420							
420					421							
421					422							
422					423							
423					424							
424					425							
425					426							
426					427							
427					428							
428					429							
429					430							
430					431							
431					432							
432					433							
433					434							
434					435							
435					436							
436					437							
437					438							
438					439							
439					440							
440					441							
441					442							
442					443							
443					444							
444					445							
445					446							
446					447							
447					448							
448					449							
449					450							
450					451							
451					452							
452					453							
453					454							
454					455							
455					456							
456					457							
457					458							
458					459							
459					460							
460					461							
461					462							
462					463							
463					464							
464					465							
465					466							
466					467							
467					468							
468					469							
469					470							
470					471							
471					472							
472					473							
473					474							
474					475							
475					476							
476					477							
477					478							
478					479							
479					480							
480					481							
48												

- 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 '\?'
- 🖱️ man 7 ascii
- 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)

8 BITOWE STRONY KODOWE

- **Strona kodowa:** kodowanie binarne znaków pisarskich
- Polskie ogonki, 8 bitowe rozszerzenia ASCII:
 - ☞ ISO 8859-2 (latin2), alfabet łaciński dla Europy środkowej i wschodniej, UNIX/GNU Linux
 - ☞ CP-1250, języki środkowoeuropejskie, MS Windows
 - ☞ CP-852, MS DOS
- ☞ Kodowanie polskich znaków diakrycznych

- **Unicode** - standard z założenia obejmujący wszystkie języki świata (obecnie ponad 110k znaków ze 100 grup językowych).
- Określa przydział przestrzeni numeracyjnej poszczególnym grupom znaków
- Kodowanie znaków: UCS (Universal Character Set), UTF (Unicode Transformation Format), UTF-8, UTF-16, UTF-32
- UTF-8 - znaki kodowane za pomocą 1, 2, 3 bajtów
- Każdy tekst w ASCII jest tekstem w UTF-8
- Znaki ASCII od 0 do 127 są mapowane jako jeden bajt.
- Znaki spoza ASCII (np. polskie znaki) są mapowane jako dwa bajty (lub więcej).
- Ten sam znak można zapisać na kilka sposobów - wybieramy najkrótszy zapis.

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

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

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

A	\0
---	----

PRZYKŁAD W C

```
#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!

```
scanf("%s",...);
```

```
#include<stdio.h>
int main()
{
    char text[10];
    scanf("%s", text);
}
```

- niebezpieczeństwo przepełnienia tablicy
- `scanf("%10s",text);`
- wczyta tylko pierwszy wyraz, nie wczyta całego łańcucha z odstępami: "Ala ma kota"

```
char *gets(char *s);
```

- funkcja `gets()` czyta linię tekstu ze standardowego wejścia i umieszcza ją w tablicy `s`
- Uwaga: brak możliwości zabezpieczenia się przez przepełnieniem tablicy

```
#include<stdio.h>
int main()
{
    char text[10];
    gets(text);
}
```

```
char *fgets(char *s, int n, FILE *f);
```

- czyta linię tekstu, ale nie więcej niż `n` znaków, ze strumienia `f` i umieszcza tekst w tablicy `s`
- obowiązkowe podanie parametru `n`
- standardowy strumień wejściowy to `stdin`

```
#include<stdio.h>
int main()
{
    char text[10];
    fgets(text, 10, stdin);
}
```

WYPISYWANIE ŁAŃCUCHA ZNAKOWEGO

```
int puts(char *s);
```

- wypisuje łańcuch i dodaje znak nowej linii

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

- wypisuje napis zgodnie z zadany formatem

Input: `printf("Color %s, Number %d, Float %5.2f", "red", 123456, 3.14;)`

Output: Color red, Number 123456, Float 3.14

SPECYFIKACJA FORMATU

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

%[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                 : %.3f \n", 3.1416);
14     printf ("   dopelnienie              : %20.3f \n", 3.1416);
15     printf ("Szerokosc pola   *         : %*d \n", 20, 123);
16     printf ("Precyzja   .*
17     printf ("Napis
18     printf ("   dopelnienie
19     printf ("   obciecie
20
21     return 0;
22 }

```

Znaki	: A A
Liczby calkowite	: 123
ze znakiem	: +123
szesnastkowo	: 7b 0x7b
dopelnienie spacjami	: 123
dopelnienie zerami	: 000000000000000000123
Zmienopozycyjne	: 3.141600
notacja naukowa	: 3.141600e+00
precyzja	: 3.142
...	

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf ("Znaki
6      printf ("Liczby calkow
7      printf ("   ze znakiem
8      printf ("   szesnastkow
9      printf ("   dopelnienie
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 }

```

```

...
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

```

 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
-

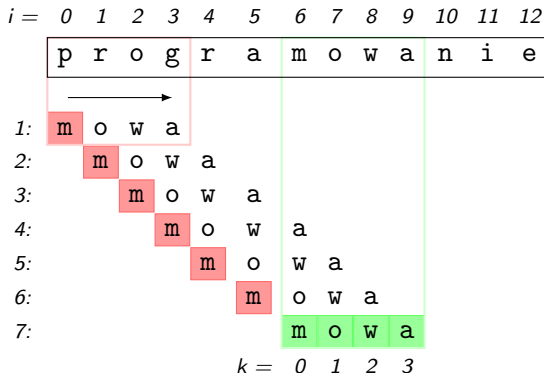
$|\mathcal{W}|$ oznacza długość łańcucha $\mathcal{W}$

```
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 }
```

ALGORYTM NAIWNY

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

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

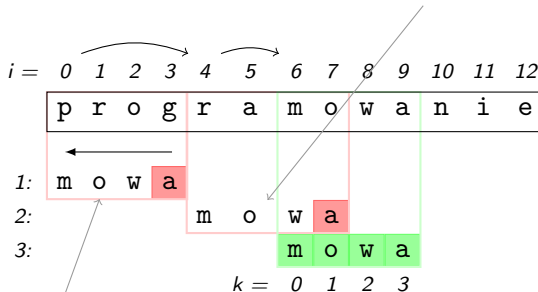


UPROSZCZONY ALGORYTM BOYERA-MOORE'A

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

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

'o' występuje we wzorcu na pozycji 1
przesunięcie okna o $k-1 = 2$



'g' nie występuje we wzorcu
przesunięcie okna o $k+1 = 4$

TABLICA PRZESUNIĘĆ

- informacja o tym czy znak występuje we wzorcu
- rozmiar tablicy odpowiada liczbie znaków danego alfabetu, dla kodu ASCII wystarczy 128 elementów
- przechowuje pozycję ostatniego wystąpienia znaku we wzorcu lub -1 gdy znak nie występuje
- $tp[w[k]] = k$
- przesunięcie okna o wartość $\max(k - tp[w[k]], 1)$

TABLICA PRZESUNIĘĆ

```
1 void utworz_tp(int tp[] , char w[])
2 {
3     int i=0;
4
5     while(i<128)
6     {
7         tp[i]=-1;
8         i=i+1;
9     }
10
11     i=0;
12     while(w[i] != '\0')
13     {
14         tp[w[i]]=i;
15         i=i+1;
16     }
17 }
```

Algorytm 2 Uproszczony algorytm Boyera-Moore'a

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

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

- 1: $\mathcal{P} \leftarrow \text{utworz_tp}(\mathcal{W})$ ▷ tworzenie tablicy przesunięć
 - 2: $i \leftarrow 0$
 - 3: **dopóki** $i \leq |\mathcal{T}| - |\mathcal{W}|$ **wykonuj**
 - 4: $k \leftarrow |\mathcal{W}|$ ▷ porównywanie od tyłu
 - 5: **dopóki** $k \geq 0$ **i** $\mathcal{T}[i + k] = \mathcal{W}[k]$ **wykonuj**
 - 6: $k \leftarrow k - 1$
 - 7: **jeżeli** $k = -1$ **wykonaj**
 - 8: **zwróć** i
 - 9: **w przeciwnym wypadku**
 - 10: $i \leftarrow i + \max(1, k - \mathcal{P}[\text{kod}(\mathcal{T}[i + k])])$
 - 11: **zwróć** -1
-

gdzie $\text{kod}(x)$ oznacza kod liczbowy znaku x .

```

1  int strindex2(char t[], char w[])
2  {
3      int i, k, z, tl, wl;
4      int tp[128];
5
6      utworz_tp(tp, w);
7      tl=strlen(t);
8      wl=strlen(w);
9
10     i=0;
11     while(i <= tl-wl)
12     {
13         k=wl-1;
14         while(k>=0 && t[i+k]==w[k] ) k=k-1;
15         if(k== -1) return i;
16         z=k-tp[t[i+k]];
17         if(z<1) z=1;
18         i = i + z;
19     }
20     return -1;
21 }

```

- Typ znakowy jest liczbą całkowitą
- Łańcuch to tablica znaków zakończona znakiem `'\0'`
- Stała znakowa w apostrofach `'A'`
- Stała napisowa w cudzysłowach `"A"` jest tablicą
- Dostęp do znaku `t[i]`
- `t == "napis"` tak nie wolno porównywać, trzeba znak po znaku

-  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