

Wykład 5: Łańcuchy znakowe.

dr inż. Andrzej Stafiniak

Wrocław 2023



HR EXCELLENCE IN RESEARCH



Politechnika Wroclawska

Podstawowe informacje o łańcuchach znakowych

- **Znak** - symbol graficzny którego znaczenie interpretowane jest w zdefiniowany sposób.
- Pogrupowane sekwencje znaków tworzą programy źródłowe, którymi możemy sterować pracą komputera. **łańcuch** to zespół znaków traktowana jako całość.
- W języku C brak jest specjalnego typu danych do reprezentacji całych napisów, dlatego napis jest traktowany jako **łańcuch znaków**.
- W języku C istnieje typ znakowy **char**. Zmienna tego typu może przechowywać wartość całkowitą (`char litera = 97`) jak i odpowiadający tej wartości znak (`char litera = 'a'`). Znakom przypisane są wartości liczbowe zgodnie z tabelą ASCII.
- Typ **char** zajmuje 1 bajt czyli 8 bitów. Standardowy kod ASCII potrzebuje 7 bitów (0-127). Rozszerzona tablica ASCII – 8 bitów (0-255).

Podstawowa tablica ASCII

Bin	Dec	Hex	Znak	Skrót
0000 0000	0	00	Null	NUL
0000 0001	1	01	Start of Heading	SOH
0000 0010	2	02	Start of Text	STX
0000 0011	3	03	End of Text	ETX
0000 0100	4	04	End of Transmission	EOT
0000 0101	5	05	Enquiry	ENQ
0000 0110	6	06	Acknowledge	ACK
0000 0111	7	07	Bell	BEL
0000 1000	8	08	Backspace	BS
0000 1001	9	09	Horizontal Tab	HT
0000 1010	10	0A	Line Feed	LF
0000 1011	11	0B	Vertical Tab	VT
0000 1100	12	0C	Form Feed	FF
0000 1101	13	0D	Carriage Return	CR
0000 1110	14	0E	Shift Out	SO
0000 1111	15	0F	Shift In	SI
0001 0000	16	10	Data Link Escape	DLE
0001 0001	17	11	Device Control 1 (XON)	DC1
0001 0010	18	12	Device Control 2	DC2
0001 0011	19	13	Device Control 3 (XOFF)	DC3
0001 0100	20	14	Device Control 4	DC4
0001 0101	21	15	Negative Acknowledge	NAK
0001 0110	22	16	Synchronous Idle	SYN
0001 0111	23	17	End of Transmission Block	ETB
0001 1000	24	18	Cancel	CAN
0001 1001	25	19	End of Medium	EM
0001 1010	26	1A	Substitute	SUB
0001 1011	27	1B	Escape	ESC
0001 1100	28	1C	File Separator	FS
0001 1101	29	1D	Group Separator	GS
0001 1110	30	1E	Record Separator	RS
0001 1111	31	1F	Unit Separator	US

Bin	Dec	Hex	Znak
0010 0000	32	20	Spacja
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	\$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Bin	Dec	Hex	Znak
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D]
0101 1110	94	5E	^
0101 1111	95	5F	_

Bin	Dec	Hex	Znak	Skrót
0110 0000	96	60	`	
0110 0001	97	61	a	
0110 0010	98	62	b	
0110 0011	99	63	c	
0110 0100	100	64	d	
0110 0101	101	65	e	
0110 0110	102	66	f	
0110 0111	103	67	g	
0110 1000	104	68	h	
0110 1001	105	69	i	
0110 1010	106	6A	j	
0110 1011	107	6B	k	
0110 1100	108	6C	l	
0110 1101	109	6D	m	
0110 1110	110	6E	n	
0110 1111	111	6F	o	
0111 0000	112	70	p	
0111 0001	113	71	q	
0111 0010	114	72	r	
0111 0011	115	73	s	
0111 0100	116	74	t	
0111 0101	117	75	u	
0111 0110	118	76	v	
0111 0111	119	77	w	
0111 1000	120	78	x	
0111 1001	121	79	y	
0111 1010	122	7A	z	
0111 1011	123	7B	{	
0111 1100	124	7C		
0111 1101	125	7D	}	
0111 1110	126	7E	~	
0111 1111	127	7F	Delete	DEL

Podstawowa tablica ASCII + Windows-1250 (CP-1250)

Bin	Dec	Hex	Znak	Skrót
0000 0000	0	00	Null	NUL
0000 0001	1	01	Start of Heading	SOH
0000 0010	2	02	Start of Text	STX
0000 0011	3	03	End of Text	ETX
0000 0100	4	04	End of Transmission	EOT
0000 0101	5	05	Enquiry	ENQ
0000 0110	6	06	Acknowledge	ACK
0000 0111	7	07	Bell	BEL
0000 1000	8	08	Backspace	BS
0000 1001	9	09	Horizontal Tab	HT
0000 1010	10	0A	Line Feed	LF
0000 1011	11	0B	Vertical Tab	VT
0000 1100	12	0C	Form Feed	FF
0000 1101	13	0D	Carriage Return	CR
0000 1110	14	0E	Shift Out	SO
0000 1111	15	0F	Shift In	SI
0001 0000	16	10	Data Link Escape	DLE
0001 0001	17	11	Device Control 1 (XON)	DC1
0001 0010	18	12	Device Control 2	DC2
0001 0011	19	13	Device Control 3 (XOFF)	DC3
0001 0100	20	14	Device Control 4	DC4
0001 0101	21	15	Negative Acknowledge	NAK
0001 0110	22	16	Synchronous Idle	SYN
0001 0111	23	17	End of Transmission Block	ETB
0001 1000	24	18	Cancel	CAN
0001 1001	25	19	End of Medium	EM
0001 1010	26	1A	Substitute	SUB
0001 1011	27	1B	Escape	ESC
0001 1100	28	1C	File Separator	FS
0001 1101	29	1D	Group Separator	GS
0001 1110	30	1E	Record Separator	RS
0001 1111	31	1F	Unit Separator	US

Bin	Dec	Hex	Znak
0010 0000	32	20	Spacja
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	\$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Bin	Dec	Hex	Znak
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D]
0101 1110	94	5E	^
0101 1111	95	5F	_

Bin	Dec	Hex	Znak	Skrót
0110 0000	96	60	`	
0110 0001	97	61	a	
0110 0010	98	62	b	
0110 0011	99	63	c	
0110 0100	100	64	d	
0110 0101	101	65	e	
0110 0110	102	66	f	
0110 0111	103	67	g	
0110 1000	104	68	h	
0110 1001	105	69	i	
0110 1010	106	6A	j	
0110 1011	107	6B	k	
0110 1100	108	6C	l	
0110 1101	109	6D	m	
0110 1110	110	6E	n	
0110 1111	111	6F	o	
0111 0000	112	70	p	
0111 0001	113	71	q	
0111 0010	114	72	r	
0111 0011	115	73	s	
0111 0100	116	74	t	
0111 0101	117	75	u	
0111 0110	118	76	v	
0111 0111	119	77	w	
0111 1000	120	78	x	
0111 1001	121	79	y	
0111 1010	122	7A	z	
0111 1011	123	7B	{	
0111 1100	124	7C		
0111 1101	125	7D	}	
0111 1110	126	7E	~	
0111 1111	127	7F	Delete	DEL

Porównanie kodowań											
Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2
0x80	€	ZK	0xA0			0xC0		Ŕ	0xE0		ŕ
0x81	NZ	ZK	0xA1	ˆ	À	0xC1		Á	0xE1		á
0x82	.	ZK	0xA2	˜		0xC2		Â	0xE2		â
0x83	NZ	ZK	0xA3		Ł	0xC3		Ã	0xE3		ã
0x84	„	ZK	0xA4		“	0xC4		Ä	0xE4		ä
0x85	...	ZK	0xA5	Å	Ĺ	0xC5		Ľ	0xE5		l
0x86	†	ZK	0xA6	‡	Š	0xC6		Č	0xE6		č
0x87	‡	ZK	0xA7		§	0xC7		Ç	0xE7		ç
0x88	NZ	ZK	0xA8	ˆ		0xC8		Ć	0xE8		ć
0x89	‰	ZK	0xA9	©	Š	0xC9		Ė	0xE9		ė
0x8A	Š	ZK	0xAA		§	0xCA		Ę	0xEA		ę
0x8B	«	ZK	0xAB	«	Ŧ	0xCB		Ė	0xEB		ė
0x8C	Š	ZK	0xAC	ˆ	Ž	0xCC		Ė	0xEC		ė
0x8D	†	ZK	0xAD			0xCD		Į	0xED		į
0x8E	Ž	ZK	0xAE	®	Ž	0xCE		Į	0xEE		į
0x8F	Ž	ZK	0xAF		Ž	0xCF		Đ	0xEF		đ
0x90	NZ	ZK	0xB0		ˆ	0xD0		Đ	0xF0		đ
0x91	ˆ	ZK	0xB1	±	ą	0xD1		Ń	0xF1		ń
0x92	ˆ	ZK	0xB2		ˆ	0xD2		Ń	0xF2		ń
0x93	ˆ	ZK	0xB3		ı	0xD3		Ō	0xF3		ó
0x94	ˆ	ZK	0xB4		ˆ	0xD4		Ō	0xF4		ó
0x95	ˆ	ZK	0xB5	μ	ı	0xD5		Ō	0xF5		ó
0x96	—	ZK	0xB6	¶	š	0xD6		Ō	0xF6		ó
0x97	—	ZK	0xB7	ˆ	ˆ	0xD7		×	0xF7		×
0x98	NZ	ZK	0xB8		ˆ	0xD8		Ŕ	0xF8		ŕ
0x99	™	ZK	0xB9	ą	š	0xD9		Ū	0xF9		ū
0x9A	š	ZK	0xBA		§	0xDA		Ū	0xFA		ū
0x9B	ˆ	ZK	0xBB	»	Ŧ	0xDB		Ū	0xFB		ū
0x9C	š	ZK	0xBC	Ł	ž	0xDC		Ū	0xFC		ū
0x9D	Ŧ	ZK	0xBD		ˆ	0xDD		Ÿ	0xFD		ÿ
0x9E	ž	ZK	0xBE	ı	ž	0xDE		Ŧ	0xFE		Ŧ
0x9F	ž	ZK	0xBF		ž	0xDF		ß	0xFF		ˆ

Windows-1250 – 8 bitowe rozszerzenie kodu ASCII zwane stronami kodowymi. Używane są przez system Windows do reprezentacji tekstów w językach środkowoeuropejskich.

www.wikipedia.org



Podstawowa tablica ASCII + Windows-1250 (CP-1250)

Bin	Dec	Hex	Znak	Skrót
0000 0000	0	00	Null	NUL
0000 0001	1	01	Start of Heading	SOH
0000 0010	2	02	Start of Text	STX
0000 0011	3	03	End of Text	ETX
0000 0100	4	04	End of Transmission	EOT
0000 0101	5	05	Enquiry	ENQ
0000 0110	6	06	Acknowledge	ACK
0000 0111	7	07	Bell	BEL
0000 1000	8	08	Backspace	BS
0000 1001	9	09	Horizontal Tab	HT
0000 1010	10	0A	Line Feed	LF
0000 1011	11	0B	Vertical Tab	VT
0000 1100	12	0C	Form Feed	FF
0000 1101	13	0D	Carriage Return	CR
0000 1110	14	0E	Shift Out	SO
0000 1111	15	0F	Shift In	SI
0001 0000	16	10	Data Link Escape	DLE
0001 0001	17	11	Device Control 1 (XON)	DC1
0001 0010	18	12	Device Control 2	DC2
0001 0011	19	13	Device Control 3 (XOFF)	DC3
0001 0100	20	14	Device Control 4	DC4
0001 0101	21	15	Negative Acknowledge	NAK
0001 0110	22	16	Synchronous Idle	SYN
0001 0111	23	17	End of Transmission Block	ETB
0001 1000	24	18	Cancel	CAN
0001 1001	25	19	End of Medium	EM
0001 1010	26	1A	Substitute	SUB
0001 1011	27	1B	Escape	ESC
0001 1100	28	1C	File Separator	FS
0001 1101	29	1D	Group Separator	GS
0001 1110	30	1E	Record Separator	RS
0001 1111	31	1F	Unit Separator	US

Bin	Dec	Hex	Znak
0010 0000	32	20	Spacja
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	\$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Bin	Dec	Hex	Znak
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D]
0101 1110	94	5E	^
0101 1111	95	5F	_

Bin	Dec	Hex	Znak	Skrót
0110 0000	96	60	`	
0110 0001	97	61	a	
0110 0010	98	62	b	
0110 0011	99	63	c	
0110 0100	100	64	d	
0110 0101	101	65	e	
0110 0110	102	66	f	
0110 0111	103	67	g	
0110 1000	104	68	h	
0110 1001	105	69	i	
0110 1010	106	6A	j	
0110 1011	107	6B	k	
0110 1100	108	6C	l	
0110 1101	109	6D	m	
0110 1110	110	6E	n	
0110 1111	111	6F	o	
0111 0000	112	70	p	
0111 0001	113	71	q	
0111 0010	114	72	r	
0111 0011	115	73	s	
0111 0100	116	74	t	
0111 0101	117	75	u	
0111 0110	118	76	v	
0111 0111	119	77	w	
0111 1000	120	78	x	
0111 1001	121	79	y	
0111 1010	122	7A	z	
0111 1011	123	7B	{	
0111 1100	124	7C		
0111 1101	125	7D	}	
0111 1110	126	7E	~	
0111 1111	127	7F	Delete	DEL

Porównanie kodowań											
Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2	Hex	CP-1250	ISO 8859-2
0x80	€	ZK	0xA0			0xC0		Ŕ	0xE0		ŕ
0x81	/Z	ZK	0xA1	·	À	0xC1		Á	0xE1		á
0x82	,	ZK	0xA2			0xC2		Â	0xE2		â
0x83	/Z	ZK	0xA3		Ĺ	0xC3		Ã	0xE3		ã
0x84	„	ZK	0xA4		▯	0xC4		Ä	0xE4		ä
0x85	...	ZK	0xA5	À	Ĺ	0xC5		Ľ	0xE5		l
0x86	†	ZK	0xA6	ı	Š	0xC6		Č	0xE6		č
0x87	‡	ZK	0xA7		Š	0xC7		Ç	0xE7		ç
0x88	/Z	ZK	0xA8			0xC8		Ć	0xE8		ć
0x89	‰	ZK	0xA9	©	Š	0xC9		Ė	0xE9		ė
0x8A	Š	ZK	0xAA		Š	0xCA		Ę	0xEA		ę
0x8B	«	ZK	0xAB	«	Ŧ	0xCB		Ė	0xEB		ė
0x8C	Š	ZK	0xAC	·	Ž	0xCC		Ė	0xEC		ė
0x8D	Ŧ	ZK	0xAD			0xCD		Į	0xED		į
0x8E	Ž	ZK	0xAE	®	Ž	0xCE		Į	0xEE		į
0x8F	Ž	ZK	0xAF		Ž	0xCF		Đ	0xEF		đ
0x90	/Z	ZK	0xB0		·	0xD0		Đ	0xF0		đ
0x91	·	ZK	0xB1	±	ą	0xD1		Ń	0xF1		ń
0x92	·	ZK	0xB2		·	0xD2		Ń	0xF2		ń
0x93	“	ZK	0xB3		ı	0xD3		Ō	0xF3		ó
0x94	”	ZK	0xB4		·	0xD4		Ō	0xF4		ó
0x95	•	ZK	0xB5	μ	ı	0xD5		Ö	0xF5		ö
0x96	—	ZK	0xB6	¶	š	0xD6		Ö	0xF6		ö
0x97	—	ZK	0xB7	·	·	0xD7		×	0xF7		×
0x98	/Z	ZK	0xB8		·	0xD8		Ř	0xF8		ř
0x99	™	ZK	0xB9	ą	š	0xD9		Ů	0xF9		ů
0x9A	š	ZK	0xBA		š	0xDA		Ů	0xFA		ů
0x9B	›	ZK	0xBB	»	ť	0xDB		Ů	0xFB		ů
0x9C	ś	ZK	0xBC	Ĺ	ž	0xDC		Ů	0xFC		ů
0x9D	ť	ZK	0xBD		·	0xDD		Ý	0xFD		ý
0x9E	ž	ZK	0xBE	ı	ž	0xDE		Ť	0xFE		ť
0x9F	ž	ZK	0xBF		ž	0xDF		ß	0xFF		·

Alternatywa - zestaw znaków **Unicode** – system kodowania **UTF-8, UTF-16, UTF-32**

www.wikipedia.org



Politechnika Wroclawska

łańcuchy znakowe

Znaki

np.: 'q' 'w' 'e' '!' '('

łańcuch znaków

"Stop war!"

Stała łańcuchowa - `#define NAPIS "Stop war!"`

- W języku C brak jest typu danych do reprezentacji **łańcuchów znaków**.
- łańcuchy przechowywane są w **tablicach** zbudowanych z elementów typu **char**.

Tablica char'ów (ale jeszcze nie tablica zawierająca łańcuch znakowy)

S	t	o	p		w	a	r	!
---	---	---	---	--	---	---	---	---

Tablica

- **Tablica** to ciąg wartości tego samego typu przechowywanych w kolejno po sobie następujących komórkach pamięci.
- Tak jak zmienne czy funkcję **tablice**, mają nazwę, czyli swój identyfikator.
- W celu dokonania deklaracji tablicy musimy użyć **typu** jaki będzie przechowywać tablica, **nazwy** tablicy oraz operatora indeksu **[]** wraz z **rozmiarem** tablicy. np.: **char tab[9]** ;
- Dostęp do składowych/elementów tablicy można uzyskać poprzez podanie **indeksu tablicy** – czyli odpowiedniego numeru komórki, pamiętając, że **tablice numerowane są od indeksu 0**.

komórka	S	t	o	p		w	a	r	!
indeks	0	1	2	3	4	5	6	7	8

Jednowymiarowa tablica char'ów

- Składnia deklaracji statycznej tablicy jednowymiarowej jest następująca:

```
typ nazwaTablicy[rozmiar];
```

```
char tab[9];    int studentIndex[6];    float ocena[NR]
```

char tab[9]; - deklaracja 9-cio elementowej tablicy char'ów

#define NR 150

```
tab[0] = 'S';  
tab[1] = 't';  
...  
tab[8] = '!';
```

Przypisanie wartości do poszczególnych komórek tablicy char'ów

~~int nr = 150;
float ocena[nr];~~

komórka	S	t	o	p		w	a	r	!
indeks	0	1	2	3	4	5	6	7	8

Dalej nie jest to tablica zawierająca łańcuch znakowy

Inicjalizacja tablicy łańcuchem znakowym

- W celu możliwości użycia łańcuchów znakowych przez różne funkcje obsługi wejścia-wyjścia, w tablicy musi być przewidziane miejsce na specjalny **znak końca łańcucha (znak zerowy) \0** - *null character*
- Do zainicjalizowania tablicy znakowej możemy wykorzystać następujące sposoby:

```
char tab[] = {'S', 't', 'o', 'p', ' ', 'w', 'a', 'r', '!', '\0'};
```

```
char tab[] = {"Stop war!"};
```

```
char tab[] = "Stop war!";
```

komórka	S	t	o	p		w	a	r	!	\0	↖ znak końca łańcucha
indeks	0	1	2	3	4	5	6	7	8	9	

Tablica zawierająca łańcuch znakowy

Inicjalizacja tablicy łańcuchem znakowym

- Przedstawione sposoby **deklaracji** oraz **inicjalizacji** tablic nie wymagają podawania **jawnie** rozmiaru tablicy - kompilator zarezerwuje odpowiednią ilość komórek.
- Podając w sposób jawny rozmiar tablicy musimy pamiętać, że rozmiar musi być co najmniej o 1 większy niż ilość znaków.

```
char tab[] = {'S', 't', 'o', 'p', ' ', 'w', 'a', 'r', '!', '\0'};
```

```
char tab[] = {"Stop war!"};
```

```
char tab[] = "Stop war!";
```

- Inicjalizując tablicę stałą łańcuchową automatycznie dodany zostanie **znak końca łańcucha** **\0**.

komórka	S	t	o	p		w	a	r	!	\0	← znak końca łańcucha
indeks	0	1	2	3	4	5	6	7	8	9	

Tablica zawierająca łańcuch znakowy

Inicjalizacja tablicy łańcuchem znakowym

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    char tab1 [] = {'H', 'e', 'l', 'l', 'o'};
```

```
    char tab2 [] = "Hello";
```

```
    printf("%s", tab1);
```

```
    printf(" - rozmiar tablicy: %d\n", sizeof tab1);
```

```
    printf("%s", tab2);
```

```
    printf(" - rozmiar tablicy: %d", sizeof tab2);
```

```
    return 0;
```

```
}
```

Brak znaku zerowego **\0**

```
Hello a - rozmiar tablicy: 5
Hello - rozmiar tablicy: 6
```

Przy braku **znaku zerowego** funkcja `printf()` odwoła się do jakiejś przypadkowej wartości

komórka

S	t	o	p		w	a	r	!	
---	---	---	---	--	---	---	---	---	--

indeks

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Funkcje obsługi znaków oraz łańcuchów znakowych

➤ Plik nagłówkowy `<ctype.h>` - funkcje obsługi znaków

Funkcja sprawdza czy znak przekazany jako argument jest:

Nazwa funkcji	Opis
<code>isalnum</code>	jest cyfrą lub literą alfabetu.
<code>isalpha</code>	jest literą alfabetu.
<code>isctrl</code>	jest znakiem kontrolnym.
<code>isdigit</code>	jest cyfrą.
<code>isgraph</code>	jest znakiem graficznym.
<code>islower</code>	jest małą literą alfabetu.
<code>isprint</code>	jest znakiem drukowalnym.
<code>ispunct</code>	jest znakiem drukowalnym ale nie jest znakiem alfanumerycznym ani spacją.
<code>isspace</code>	jest białym znakiem.
<code>isupper</code>	jest dużą literą alfabetu.
<code>isxdigit</code>	jest cyfrą szesnastkową.
<code>tolower</code>	zwraca znak zamieniony z dużej litery na małą.
<code>toupper</code>	zwraca znak zamieniony z małej litery na dużą.

```
int isalpha(int ch);
```

- funkcja ta zwraca wartość **różną od zera** gdy argument, który został przekazany do funkcji **jest** znakiem alfabetu.

```
char znak;  
scanf("%c", &znak);  
if(isalpha(znak))  
    printf( „%c to litera  
            alfabetu”, znak);  
else  
    printf( „%c nie należy  
            do alfabetu”, znak);
```

Funkcje obsługi znaków oraz łańcuchów znakowych

➤ Plik nagłówkowy **<string.h>** - funkcje obsługi łańcuchów znakowych

Nazwa funkcji	Opis
atol	Konwertuje wartość zapisaną w łańcuchu znaków do postaci liczby typu całkowitego (long).
strcat	Scala dwa łańcuchy znaków w jeden.
strchr	Szuka pierwszego wystąpienia znaku w łańcuchu znaków.
strcmp	Porównuje dwa łańcuchy znaków.
strcpy	Kopiuje łańcuch znaków do tablicy znaków.
strcspn	Szuka pierwszego wystąpienia znaku (z puli znaków) w łańcuchu znaków.
strerror	Zwraca łańcuch znaków zawierający komunikat błędu dla wskazanego numeru błędu.
stricmp	Porównuje dwa łańcuchy znaków (ignoruje wielość liter).
strlen	Oblicza długość łańcucha.
strncat	Scala dwa łańcuchy znaków w jeden. Uwzględnia maksymalną liczbę znaków, jaka może zostać dopisana.
strncmp	Porównuje określoną liczbę znaków dwóch łańcuchów znaków.
strncpy	Kopiuje określoną liczbę znaków łańcucha.
strrchr	Szuka ostatniego wystąpienia znaku w łańcuchu znaków.
strspn	Zwraca indeks pierwszego znaku, który nie należy do puli znaków.
strstr	Szuka pierwszego wystąpienia łańcucha znaków w innym łańcuchu znaków.
strtok	Zastępuje (w łańcuchu znaków) pierwszy znaleziony znak znakiem terminalnym.
strtol	Konwertuje wartość zapisaną w łańcuchu znaków w dowolnym systemie liczbowym do postaci liczby typu całkowitego (long).

Funkcje obsługi łańcuchów znakowych

strlen	Oblicza długość łańcucha znaków.
---------------	----------------------------------

Nagłówek funkcji `size_t strlen (const char *str)`

Typ zwracany, liczby całkowite
dodatnie `unsigned int`

Wskaźnik na pierwszy element tablicy w
której przechowywany jest łańcuch

Funkcja zwraca ilość znaków w łańcuchu do napotkania znaku zerowego `\0`. W przypadku ciągów niezakończonych znakiem zerowym działanie funkcji jest nieprzewidziane.

```
char tab[] = "No war, no more!";  
int counter;  
counter = strlen (tab);
```

Jaka wartość zostanie przypisana
do zmiennej `counter` ?

Funkcje obsługi łańcuchów znakowych

strlen Oblicza długość łańcucha znaków.

Nagłówek funkcji `size_t strlen (const char *str)`

```
#include <stdio.h>
#include <string.h>

int main()
{
    char tab1 [] = {'H', 'e', 'l', 'l', 'o'};
    char tab2 [] = "Hello";
    char tab3 [7] = "Hello";

    printf("\n%s", tab1);
    printf(" - dlugosci lancucha znakow: %d\n", strlen(tab1));

    printf("%s", tab2);
    printf(" - dlugosci lancucha znakow: %d\n", strlen(tab2));

    printf("%s", tab3);
    printf(" - dlugosci lancucha znakow: %d\n", strlen(tab3));
    return 0;
}
```

```
Hello? a - dlugosci lancucha znakow: 8
Hello - dlugosci lancucha znakow: 5
Hello - dlugosci lancucha znakow: 5
```

Funkcje obsługi łańcuchów znakowych

strcmp	Porównuje dwa łańcuchy znaków.
stricmp	Porównuje dwa łańcuchy znaków (ignoruje wielość liter).
strncmp	Porównuje określoną liczbę znaków dwóch łańcuchów znaków.

Nagłówek funkcji `int strcmp (const char *str1, const char *str2)`

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
```

```
    char tab1 [] = {'H', 'e', 'l', 'l', 'o'};
    char tab2 [] = "Hello";
    char tab3 [7] = "Hello";
```

```
    printf("\ntab1: %s, tab2: %s, tab3: %s\n", tab1, tab2, tab3);
```

```
    printf("strcmp (tab1, tab2): %d\n", strcmp(tab1,tab2));
```

```
    printf("strcmp (tab2, tab3): %d\n", strcmp(tab2,tab3));
```

```
    return 0;
```

```
}
```

```
tab1: Hello? a, tab2: Hello, tab3: Hello
strcmp (tab1, tab2): 1
strcmp (tab2, tab3): 0
```

- Funkcja `strcmp` przeszukuje łańcuchy do momentu znalezienia pierwszej pary różnych znaków. Zwraca 1 - gdy pierwszy łańcuch jest mniejszy od drugiego oraz -1 w przeciwnym wypadku. Gdy łańcuchy są takie same zwraca 0.

Funkcje obsługi łańcuchów znakowych

strcat	Scala dwa łańcuchy znaków w jeden.
strncat	Scala dwa łańcuchy znaków w jeden. Uwzględnia maksymalną liczbę znaków, jaka może zostać dopisana.

Nagłówek funkcji `char strcat (char *destination, const char *source)`
`char strncat (char *destination, const char *source, size_t n)`

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
```

```
    char tab2 [] = " musi";
    char tab3 [] = " byc";
    char tab4 [] = " zainicjalizowana";
    char tab5 [100] = "Tablica wynikowa";
    char tab6 [100] = "Oraz mozna okreslic ile chcemy dolaczyc elementow do nowej ";
```

```
    strcat (tab5, tab2);
    strcat (tab5, tab3);
    strcat (tab5, tab4);
    strcat (tab5, " oraz musi byc statycznie zadeklarowany rozmiar.");
    strncat (tab6, tab5, 16);
```

```
    printf("\ntab5: %s\n", tab5);
    printf("\ntab6: %s\n", tab6);
```

```
    return 0;
```

```
}
```

tab5: Tablica wynikowa musi byc zainicjalizowana oraz musi byc statycznie zadeklarowany rozmiar.
tab6: Oraz mozna okreslic ile chcemy dolaczyc elementow do nowej Tablica wynikowa

➤ Do nas należy weryfikacja czy łańcuch dołączany zmieści się w tablicy pierwszego łańcucha.

➤ Pierwszy znak drugiego łańcucha nadpisuje znak `\0` pierwszego łańcucha. Na koniec nowego łańcucha dodawany jest znak zerowy.

Funkcje obsługi łańcuchów znakowych

strcpy	Kopiuje łańcuch znaków do tablicy znaków.
strncpy	Kopiuje określoną liczbę znaków łańcucha.

Nagłówek funkcji `char strcpy (char *destination, const char *source)`
`char strncpy (char *destination, const char *source, size_t n)`

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
    char tab1 [] = "Tekst do kopiowania";
    char tab2 [100];
    char tab3 [100];

    strcpy (tab2, tab1);
    strcpy (tab3, "Inny tekst do kopiowania1");

    printf("\ntab2: %s\n", tab2);
    printf("\ntab3: %s\n", tab3);

    strncpy (tab3, tab1, 5);
    printf("\ntab3: %s\n", tab3);

    return 0;
}
```

- Funkcje `strcpy` i `strncpy` kopiują łańcuch *source* do łańcucha *destination* do momentu napotkania znaku `\0` lub przekopiowania *n* znaków.
- Jeśli kopiowany łańcuch jest krótszy od *n*, to zostanie skopiowany cały ze znakiem `\0`. W przeciwnym wypadku łańcuch końcowy może nie mieć znaku zerowego!
- Do nas należy weryfikacja czy łańcuch kopiowany zmieści się w tablicy pierwszego łańcucha.

```
tab2: Tekst do kopiowania
tab3: Inny tekst do kopiowania1
tab3: Teksttekst do kopiowania1
```

Funkcje obsługi łańcuchów znakowych

strchr Szuka pierwszego wystąpienia znaku w łańcuchu znaków.

Nagłówek funkcji `char *strchr (const char *str, int c)`

Zwracany typ to wskaźnik czyli adres do znalezionej znaku

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
    char polecenie [] = "Zachowaj odstęp!";
    char *str;
    int c = 'a';
    str = strchr(polecenie, c);

    printf("\nstr: %d\n", str);

    if (str!=NULL){
        printf("\nstr: %d\n", *str);
        printf("\nn: %d\n", str-polecenie+1);
    }

    printf("\nPrzeszukaj cały lancuch", str-polecenie+1);
    while (str!=NULL){
        printf ("\nn: %d\n",str-polecenie+1);
        str=strchr(str+1, c);
    }

    return 0;
}
```

```
str: 6422279
str: 97
n: 2
Przeszukaj cały lancuch
n: 2
n: 7
```

Funkcje obsługi łańcuchów znakowych

strtol

Konwertuje wartość zapisaną w łańcuchu znaków w dowolnym systemie liczbowym do postaci liczby typu całkowitego (long).

Nagłówek funkcji `long strtol(const char *str, char **end, int base)`

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
    char tab [] = "321 ffff 1111";
    char *koniecLiczby;
    long int liczba1, liczba2, liczba3;

    liczba1 = strtol (tab,&koniecLiczby, 10);
    liczba2 = strtol (koniecLiczby,&koniecLiczby, 16);
    liczba3 = strtol (koniecLiczby,&koniecLiczby, 2);

    printf("\nL1: %d, L2: %d, L3: %d\n", liczba1, liczba2, liczba3);

    return 0;
}
```

Wskaźnik na wskaźnik,
czyli adres wskaźnika.

Funkcja przerywa proces konwersji w
chwili napotkania znaku, który nie
jest częścią liczby.

```
L1: 321, L2: 65535, L3: 15
```

Funkcje obsługi łańcuchów znakowych

atoi	Konwertuje wartość zapisaną w łańcuchu znaków do postaci liczby typu całkowitego (long).
-------------	--

Nagłówek funkcji **long** atoi (**const char** *str)

W pliku nagłówkowy **stdlib.h** zdefiniowane są jeszcze: **atoi()**, **atof()**.

```
#include <stdio.h>
#include <string.h>

int main() {
    long liczba;
    char tab[] = "  321abc";
    char tab1[] = " a 321abc";

    liczba = atoi(tab);
    printf("L1 = %d\n",liczba);

    liczba = atoi(tab1);
    printf("L1 = %d\n",liczba);

    return 0;
}
```

- Funkcje te przerywają proces konwersji w momencie napotkania pierwszego znaku, który nie należy do liczby.
- Początkowe znaki białe są pomijane.

```
L1: 321
L1: 0
```

Funkcje obsługi łańcuchów znakowych

Pliku nagłówkowy `stdlib.h`

Nagłówek funkcji `double strtod (const char *str, char **end)`

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char tab [] = "123.123 69.96";
    char *koniecLiczby;
    double L1, L2;

    L1 = strtod (tab, &koniecLiczby);
    L2 = strtod (koniecLiczby, &koniecLiczby);

    printf("\nL1: %f\n \nL2: %f\n", L1, L2);

    return 0;
}
```

➤ Do obsługi systemu dziesiętnego.

```
L1: 123.123000
L2: 69.960000
```

Biblioteka standardowa języka C

- **Biblioteka standardowa** języka C dostarcza zestaw funkcji ułatwiających pisanie kodów źródłowych.
- Plik nagłówkowe **biblioteki standardowej** języka C:

<code><assert.h></code>	<code><complex.h></code>	<code><ctype.h></code>
<code><errno.h></code>	<code><fenv.h></code>	<code><float.h></code>
<code><inttypes.h></code>	<code><iso646.h></code>	<code><limits.h></code>
<code><locale.h></code>	<code><math.h></code>	<code><setjmp.h></code>
<code><signal.h></code>	<code><stdalign.h></code>	<code><stdarg.h></code>
<code><stdatomic.h></code>	<code><stdbool.h></code>	<code><stddef.h></code>
<code><stdint.h></code>	<code><stdio.h></code>	<code><stdlib.h></code>
<code><stdnoreturn.h></code>	<code><string.h></code>	<code><tgmath.h></code>
<code><threads.h></code>	<code><time.h></code>	<code><uchar.h></code>
<code><wchar.h></code>	<code><wctype.h></code>	