Politechnika Białostocka

Wydział Elektryczny

Katedra Elektrotechniki Teoretycznej i Metrologii

Instrukcja do pracowni specjalistycznej

Temat ćwiczenia:

JĘZYK C - ŁAŃCUCHY ZNAKÓW

Ćwiczenie nr INF_D07

Pracownia specjalistyczna z przedmiotu:

Informatyka

Kod: **EDS1B 1007**

Opracował:

dr inż. Jarosław Forenc

Białystok 2018

Materiały zostały opracowane w ramach projektu „PB2020 – Zintegrowany Program Rozwoju Politechniki Białostockiej" realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Spis treści

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows (XP/7/10).

1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Microsoft Visual Studio 2008 Standard Edition lub Microsoft Visual Studio 2008 Express Edition zawierające kompilator Microsoft Visual C++ 2008.

2. Wiadomości teoretyczne

2.1. Deklaracja łańcucha znaków

Łańcuch znaków (napis, stała napisowa, C-string) jest to ciąg złożony z zera lub większej liczby znaków zawartych między znakami cudzysłowu, np.

```
"Program w C"
```

Znaki cudzysłowu nie są częścią napisu, służą jedynie do określenia jego granic.

Łańcuchy znaków przechowywane są w postaci tablicy, której elementami są pojedyncze znaki (tablica elementów typu **char**). Ostatnim elementem tablicy jest znak o kodzie **0** (stała liczbowa **0** lub stała znakowa **'\0'**), oznaczający koniec napisu (Rys. 1). W większości przypadków znak ten jest dodawany automatycznie.

0	1	2	3	4	5	6	7	8	9	10	11
P	r	o	g	r	a	m		w		C	\0

Rys. 1. Tablica przechowująca tekst „Program w C”

W rzeczywistości w tablicy zamiast znaków przechowywane są odpowiadające im kody ASCII czyli liczby (Rys. 2).

0	1	2	3	4	5	6	7	8	9	10	11
80	114	111	103	114	97	109	32	119	32	67	0

Rys. 2. Reprezentacja znaków w tablicy

Deklaracja zmiennej mogącej przechowywać napis jest podobna do deklaracji zwykłej tablicy:

```
char nazwa[rozmiar];
```

Rozmiar fizycznej pamięci przeznaczonej na napis musi być o jeden większy niż liczba znaków zawartych między znakami cudzysłowu. Deklaracja tablicy, w której można przechowywać napisy o maksymalnej długości do 20 znaków ma postać:

```
char txt[21];
```

2.2. Inicjalizacja łańcucha znaków

Deklarując łańcuch znaków możemy nadać mu wartość początkową, np.

```
char txt[10] = "napis";
```

Pozostałym elementom tablicy automatycznie przypisywana jest stała liczbowa **0** czyli stała znakowa **'\0'** (Rys. 3).

0	1	2	3	4	5	6	7	8	9
n	a	p	i	s	\0	\0	\0	\0	\0

Rys. 3. Tablica znaków po inicjalizacji

Inicjalizując łańcuch znaków można również podać pojedyncze znaki umieszczone w apostrofach, np.

```
char txt[10] = {'n','a','p','i','s'};
```

lub odpowiadające im kody ASCII:

```
char txt[10] = {110,97,112,105,115};
```

Deklarując napis można nie określać jego długości, kompilator przydzieli wtedy automatycznie odpowiedni rozmiar pamięci (uwzględniając ostatni znak **'\0'**):

```
char *txt = "napis";
```

lub

```
char txt[] = "napis";
```

W powyższy sposób można nadawać wartość łańcuchowi znaków tylko przy jego deklaracji. Zatem błędny jest poniższy zapis:

```
char txt[10];
txt = "napis";
```

Prawidłowy zapis wymaga wykorzystania funkcji **strcpy()** z pliku nagłówkowego **string.h**:

```
char txt[10];
strcpy(txt,"napis");
```

Funkcje znajdujące się w tym pliku nagłówkowym zostały opisane w dalszej części instrukcji (Rozdział 2.5).

2.3. Stała znakowa

Stała znakowa jest to liczba całkowita. Taką stałą tworzy jeden znak ujęty w apostrofy, np. **'x'**. Wartością stałej znakowej jest wartość kodu ASCII (Tabela 1).

Tabela 1. Wybrane kody ASCII

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

Pewne znaki niegraficzne mogą być reprezentowane w stałych znakowych przez sekwencje specjalne, które wyglądają jak dwa znaki, ale reprezentują tylko jeden znak. Należą do nich:

'\n'	nowy wiersz	**'\\'**	\ (ang. backslash)
'\t'	tabulator poziomy	**'\''**	apostrof
'\v'	tabulator pionowy	**'\"'**	cudzysłów
'\a'	alarm	**'\?'**	znak zapytania

Uwaga: zapis **'A'** oznacza jeden znak, natomiast zapis **"A"** - dwa znaki, gdyż jest to napis kończący się znakiem **'\0'**.

2.4. Funkcje do wyprowadzania i wprowadzania znaków

`printf()`	Nagłówek: **int printf(const char *format,...);**

- w funkcji **printf()** do wyświetlenia łańcucha znaków używamy specyfikatora formatu **%s**, zaś do wyświetlenia pojedynczego znaku - **%c**;

`puts()`	Nagłówek: **int puts(const char *s);**

- funkcja **puts()** wypisuje na **stdout** (**ekran**) zawartość łańcucha znakowego (ciąg znaków zakończony znakiem **'\0'**), zastępując znak **'\0'** znakiem **'\n'**.

Łańcuch znaków jest zwykłą tablicą - można więc odwoływać się do jej pojedynczych elementów. Wyświetlając te elementy przy zastosowaniu specyfikatora formatu **%c** otrzymamy znaki, zaś stosując specyfikator **%d** - odpowiadające im kody ASCII (liczby).

Program przedstawiający różne sposoby wyświetlenia łańcucha znaków.

```
include <stdio.h>

int main(void)
{
    char txt[14] = "To jest napis";

    printf(txt);          printf("\n");
    printf("%s",txt);     printf("\n");
    printf("%s\n",txt);
    puts(txt);

    printf("Znaki: ");
    for (int i=0; i<14; i++) printf("%c ",txt[i]);
    printf("\n");

    printf("Kody:  ");
    for (int i=0; i<14; i++) printf("%d ",txt[i]);
    printf("\n");

    return 0;
}
```


Wynik uruchomienia programu:

```
To jest napis
To jest napis
To jest napis
To jest napis
Znaki: T o   j e s t   n a p i s
Kody:  84 111 32 106 101 115 116 32 110 97 112 105 115 0
```

`scanf()`	Nagłówek: **int scanf(const char *format,...);**

- w funkcji **scanf()** do wczytania łańcucha znaków używamy specyfikatora formatu **%s**, zaś do wczytania pojedynczego znaku - **%c**;

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

- funkcja **gets()** wprowadza wiersz (ciąg znaków zakończony **'\n'**) ze strumienia **stdin** (klawiatura) i umieszcza w obszarze pamięci wskazywanym przez wskaźnik **s** zastępując **'\n'** znakiem **'\0'**.

  Wczytanie tekstu funkcją **scanf()** ma następującą postać:

```
char txt[15];
scanf("%s",txt);
```

Zmienna **txt** jest tablicą. Nazwa tablicy jest adresem jej początku w pamięci komputera. Z tego względu przed **txt** nie występuje znak **&**.

Funkcja **scanf()** kończy wczytywanie danych po wystąpieniu pierwszego tzw. białego znaku (spacja, tabulacja, enter). Jeśli w powyższym przykładzie użytkownik wprowadzi tekst: **"To jest napis"**, to **scanf()** zapamięta tylko pierwszy wyraz: **"To"**. Zapamiętanie całego wiersza tekstu (do naciśnięcia klawisza Enter) wymaga użycia funkcji **gets()**:

```
char txt[15];
gets(txt);
```

2.5. Funkcje z pliku nagłówkowego string.h

Plik nagłówkowy **string.h** definiuje funkcje do wykonywania operacji na łańcuchach znaków i tablicach.

`strlen()`	Nagłówek: **size_t strlen(const char *s);**

- funkcja **strlen()** zwraca długość łańcucha znaków **s**;
- nie bierze pod uwagę znaku **'\0'**;

`strcpy()`	Nagłówek: **char *strcpy(char *s1, const char *s2);**

- funkcja **strcpy()** kopiuje łańcuch **s2** do łańcucha **s1**;

`strcat()`	Nagłówek: **char *strcat(char *s1, const char *s2);**

- funkcja **strcat()** dołącza do łańcucha **s1** łańcuch **s2**;

`strchr()`	Nagłówek: **char *strchr(const char *s, int c);**

- funkcja **strchr()** przeszukuje łańcuch **s** w celu znalezienia pierwszego wystąpienia znaku **c**;
- zwraca wskaźnik do znalezionego znaku lub **NULL**, jeśli znak nie został znaleziony;

`strcmp()`	Nagłówek: **int strcmp(const char *s1, const char *s2);**

- funkcja **strcmp()** porównuje łańcuchy **s1** i **s2** z rozróżnianiem wielkości liter;
- zwraca **0**, gdy **s1=s2**, wartość mniejszą od zera, gdy **s1<s2** i wartość większą od zera, gdy **s1>s2**;

`strcmpi()`	Nagłówek: **int strcmpi(const char *s1, const char *s2);**

- funkcja **strcmpi()** działa jak **strcmp()**, ale bez rozróżniania wielkości liter;

`strlwr()`	Nagłówek: **char *strlwr(char *s);**

- funkcja **strlwr()** zamienia w łańcuchu **s** wielkie litery na małe;

`strrev()`	Nagłówek: **char *strrev(char *s);**

- funkcja **strrev()** odwraca kolejność znaków w łańcuchu **s**;

`strset()`	Nagłówek: **char *strset(char *s, int c);**

- funkcja **strset()** wypełnia łańcuch **s** znakiem **c**;

`strstr()`	Nagłówek: **char *strstr(const char *s1, const char *s2);**

- funkcja **strstr()** przeszukuje łańcuch **s1** w celu odnalezienia podłańcucha **s2** zwracając wskazanie do elementu łańcucha **s1**, od którego zaczynają się znaki takie same jak w łańcuchu **s2**, lub **NULL**, gdy **s2** nie występuje w **s1**;

`strupr()`	Nagłówek: **char *strupr(char *s);**

- funkcja **strupr()** zamienia w łańcuchu **s** litery małe na wielkie.

Poniższy program pokazuje przykładowe wykorzystanie wybranych funkcji z pliku nagłówkowego **string.h**.

Przykładowe operacje na łańcuchu znaków.

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

int main(void)
{
    char napis1[] = "Tekst w buforze";
    char napis2[20];
    int  dlugosc;
```

```
    printf("napis1: %s\n",napis1);
    dlugosc = strlen(napis1);
    printf("liczba znakow w napis1: %d\n",dlugosc);
    strupr(napis1);
    printf("napis1 (wielkie litery): %s\n",napis1);
    strlwr(napis1);
    printf("napis1 (male litery): %s\n",napis1);
    strcpy(napis2,napis1);
    printf("napis2: %s\n",napis2);
    strrev(napis2);
    printf("napis2 (odwrocony): %s\n",napis2);

    return 0;
}
```

W wyniku uruchomienia programu na ekranie pojawi się:

```
napis1: Tekst w buforze
liczba znakow w napis1: 15
napis1 (wielkie litery): TEKST W BUFORZE
napis1 (male litery): tekst w buforze
napis2: tekst w buforze
napis2 (odwrocony): ezrofub w tsket
```

2.6. Dwuwymiarowa tablica elementów typu char

Szczególnym przypadkiem tablicy elementów typu **char** jest tablica dwuwymiarowa. Używając dwóch indeksów (numeru wiersza i numeru kolumny) można odwoływać się do jej pojedynczych elementów (znaków). Jeśli natomiast użyjemy jednego indeksu (numeru wiersza), to cały wiersz zostanie potraktowany jako łańcuch znaków (napis).

W poniższym fragmencie programu znajduje się deklaracja tablicy **txt** połączona z inicjalizacją. Następnie w pętli **for** wyświetlane są kolejne wiersze tej tablicy (**txt[i]**).

```
char txt[3][15] = {"Programowanie",
                   "nie jest",
                   "trudne"};
```



```
for (int i=0; i<3; i++)
    printf("%s ",txt[i]);
printf("\n");
```

Wynik uruchomienia powyższego kodu:

```
Programowanie nie jest trudne
```

Każdy wiersz tablicy ma jednakową długość (**15**), ale przechowuje tekst o różnej liczbie znaków (Rys. 4).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	P	r	o	g	r	a	m	o	w	a	n	i	e	\0	\0
1	n	i	e		j	e	s	t	\0	\0	\0	\0	\0	\0	\0
2	t	r	u	d	n	e	\0	\0	\0	\0	\0	\0	\0	\0	\0

Rys. 4. Dwuwymiarowa tablica przechowująca łańcuchy znaków

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Napisz program wyświetlający na ekranie kolejne **kody** ASCII (od **30** do **254**) oraz odpowiadające im **znaki**. Przykładowy fragment wydruku:

```
 91 - [  92 - \  93 - ]  94 - ^  95 - _  96 - `  97 - a
 98 - b  99 - c 101 - e 102 - f 103 - g 104 - h 105 - i
106 - j 107 - k 108 - l 109 - m 111 - o 112 - p 113 - q
```

2. Napisz program, który wczytuje z klawiatury jeden wiersz tekstu, a następnie:
   - wyświetla na ekranie wczytany tekst;
   - oblicza i wyświetla liczbę znaków znajdujących się w tekście;
   - oblicza i wyświetla liczbę małych liter oraz liczbę wielkich liter znajdujących się w tekście;

- zamienia wszystkie małe litery na wielkie i ponownie wyświetla tekst;
- zamienia wszystkie wielkie litery na małe i ponownie wyświetla tekst.

Uwaga: nie stosuj funkcji z pliku nagłówkowego **string.h**.

3. Napisz program, w którym użytkownik wczytuje z klawiatury trzy liczby całkowite: **h**, **m**, **s** (**h** - godzina, **m** - minuta, **s** - sekunda). Następnie program powinien zapisać do tablicy łańcuch znaków zawierający czas w formacie **hh:mm:ss** (np. **15:05:30**). Wyświetl zawartość tablicy na ekranie (Uwaga: do rozwiązania zadania wykorzystaj funkcję **sprintf()**).

4. Napisz program, który wczytuje tekst z klawiatury (jeden wiersz), a następnie usuwa wszystkie znaki spacji znajdujące się na początku tekstu i na jego końcu, np.

   `"   Ala ma kota   " → "Ala ma kota"`

5. Napisz program wczytujący z klawiatury do 10 linii tekstu. Wczytywanie zostaje zakończone po wczytaniu 10 linii lub po wprowadzeniu pustego łańcucha (naciśnięcie klawisza ENTER). Wyświetl wczytane linie w kolejności od ostatniej do pierwszej.

6. Napisz program, który będzie wczytywał ciąg znaków składający się z zer i jedynek. Następnie program powinien wyznaczyć liczbę serii w ciągu. Seria w ciągu, to podciąg złożony z jednego lub kilku takich samych znaków.

   Przykładowe wywołanie programu:

```
Podaj ciag:   0011101010011
Liczba serii: 8
```

7. Napisz program, który wczytuje jeden wiersz tekstu, a następnie podaje liczbę samogłosek występujących w tym wierszu.

8. Napisz program, który wczytuje jeden wiersz tekstu, a następnie podaje ile w tym wierszu występuje wyrazów.

4. Literatura

[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[3] Prinz P., Crawford T.: Język C w pigułce. APN Promise, Warszawa, 2016.

[4] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[5] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[6] http://www.cplusplus.com/reference/clibrary - C library - C++ Reference

5. Pytania kontrolne

1. Opisz sposób przechowywania napisów (tekstów) w języku C.
2. Przedstaw sposoby inicjalizacji tablicy znaków.
3. Scharakteryzuj funkcje znajdujące się w pliku nagłówkowym **string.h**.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.

- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.