Laboratorium 6: „Łańcuchy znaków”

dr inż. Leszek Ciopiński
dr inż. Arkadiusz Chrobot
dr inż. Grzegorz Łukawski

4 września 2026

1. Wprowadzenie

Instrukcja poświęcona jest zmiennym, które służą do przechowywania łańcuchów znaków w tym języku. Jej drugi rozdział traktuje o funkcjach umożliwiających wykonywanie operacji na tych łańcuchach.

2. Łańcuchy znaków

Język C nie posiada osobnego typu o nazwie „string". Zamiast tego pozwala on przechowywać ciągi (łańcuchy) znaków w tablicach elementów typu `char`. Należy pamiętać, że taka tablica musi mieć o jeden element więcej niż jest znaków w łańcuchu, który chcemy w niej przechować. Dodatkowe miejsce jest potrzebne do zapamiętania znaku o kodzie ASCII[1] równym zero, zapisywanym w języku C jako `'\0'`, który oznacza koniec ciągu znaków. Takie tablice mogą być tworzone jako tablice zainicjowane. Listing 1 zawiera kilka przykładów.

```
char ciag[] = {'P','r','z','y','k','ł','a','d','\0'};
char ciag1[] = "Przykład"; // Tekst podany w cudzysłowie jest traktowany jako
                           // łańcuch znaków i nie wymaga podawania znaku
                           // końca ciągu.
char ciag2[10]; // Niezainicjowana tablica znaków, która może pomieścić łańcuchy
                // o długości maksymalnie dziewięciu znaków.
```

Listing 1: Tablice znaków

2.1. Operacje na łańcuchach

Tablice w języku C są zrealizowane jako wskaźniki na obszary pamięci mogące pomieścić określoną liczbę elementów danego typu. W tej instrukcji ta cecha tablic nie będzie głębiej analizowana, jednakże ma ona pewne konsekwencje, które są szczególnie widoczne w przypadku tablic znaków. Do każdego elementu takiej tablic możemy dostać się tak samo jak do elementu w zwykłej tablicy, ale nie możemy bezpośrednio, za pomocą operator `=` przypisać jednej tablicy do drugiej lub porównywać ich za pomocą operatora `==`[2]. Standardowa biblioteka języka C dostarcza funkcji, które umożliwiają wykonanie takich operacji. Zacznijmy jednak od wczytania ciągu znaków z klawiatury i wypisania go na ekran. Możemy wydrukować łańcuch znaków na ekranie za pomocą funkcji `puts()` lub `printf()`. Listing 2 zawiera trzy przykłady. W funkcji `printf()` do wypisania zawartości zmiennej będącej tablicą znaków został użyty

```
puts("Ala ma kota.");
printf("Ala ma kota.\n");
char ciag[] = "Ala ma kota.";
puts(ciag);
printf("%s\n",ciag);
```

Listing 2: Wypisanie łańcucha znaków na ekran

ciąg formatujący "%s". Tego samego ciągu można użyć do wprowadzania łańcucha z klawiatury przy pomocy funkcji `scanf()`, ale tak użyta funkcja `scanf()` będzie wczytywała ciąg do napotkania pierwszego znaku białego (np. w przypadku ciągu "Ala ma kota." wczyta tylko "Ala"). Aby wczytać cały ciąg, należy użyć wyrażeń regularnych i przekazać funkcji, że ma czytać wszystkie znaki oprócz znaku końca wiersza ('\n'), który generowany jest przez klawisz `Enter`. Można też użyć w tym celu funkcji `fgets()`[3]. Przyjmuje ona trzy argumenty wywołania: tablicę znaków, maksymalny rozmiar wprowadzanego łańcucha

[1] Język C umożliwia również stosowanie standardów UTF, ale ten temat nie będzie tu opisywany.

[2] Dokładniej rzecz ujmując, można za pomocą tego operatora porównywać jedynie *adresy* takich tablic, ale nie ich zawartość.

[3] Dostępna jest także funkcja `gets()`, ale jej użycie nie jest zalecane.

oraz zmienną `stdin` będącą strumieniem standardowego wejścia, czyli połączeniem z klawiaturą. Listing 3 zawiera odpowiednie przykłady. Należy zauważyć, że w funkcji `scanf()` drugi argument (zmienna

```
char str[40];

scanf("%s",str); // Wczytanie ciągu do pierwszego napotkanego znaku białego.
while(getchar()!='\n'); // Usunięcie znaku \n ze strumienia wejściowego.
scanf("%[^\n]s",str); // Wczytanie całego ciągu znaków.
while(getchar()!='\n'); // Usunięcie znaku \n ze strumienia wejściowego.
fgets(str,40,stdin); // Wczytanie całego ciągu znaków.
```

Listing 3: Wprowadzanie ciągu znaków z klawiatury

`string`) *nie jest* poprzedzona operatorem, który zwraca jej adres (znak &). Dzieje się tak, ponieważ jako drugi argument wywołania ta funkcja przyjmuje adres zmiennej do której ma zapisać wartość, a ponieważ nazwa tablicy sama w sobie jest wskaźnikiem, który zawiera taki adres, więc nie należy używać operatora & w stosunku do niej. Zapis [^\n] informuje funkcję `scanf()`, że ma akceptować wszystkie znaki poza znakiem nowego wiersza. W nawiasach kwadratowych umieszczany jest zbiór znaków, które mają być zaakceptowane, a znak „^” oznacza „wszystkie, oprócz …”. Użyta w przykładzie funkcja `getchar()` odczytuje z klawiatury pojedynczy znak i zwraca jego kod ASCII po zamianie na liczbę typu `int`.

Tabela 1: Wybrane funkcje z pliku `string.h`

Przykład użycia	**Opis działania**
`int dl = strlen(str);`	Zwraca długość ciągu znaków zapisanego w zmiennej przekazanej jej przez parametr. Nie uwzględnia znaku końca ciągu, czyli `'\0'`.
`int cmp = strcmp(str1,str1);`	Porównuje dwa łańcuchy znaków. Może być używana w instrukcjach warunkowych i pętlach `while` oraz `do …while`. Zwraca wartość zero jeśli łańcuchy są identyczne, wartość mniejszą od zera, jeśli łańcuch pierwszy jest „mniejszy” od drugiego, lub dodatnią jeśli jest odwrotnie.
`int cmp = strncmp(str1,str2,n);`	Jak wyżej, ale porównuje tylko `n` pierwszych znaków. Z tego względu jest uważana za bezpieczniejszą w użyciu niż `strcmp()`.
`strcpy(str1,str2);`	Kopiuje łańcuch będący drugim argumentem jej wywołania do zmiennej będącej pierwszym argumentem wywołania. Zmienna ta powinna być wystarczająco pojemna, żeby pomieścić cały łańcuch wraz ze znakiem końca. Funkcja zwraca wskaźnik do skopiowanego łańcucha, ale najczęściej ta wartość jest ignorowana.
`strncpy(str1,str2,n);`	Jak wyżej, ale ogranicza kopiowanie do `n` początkowych znaków.
`strcat(str1,str2);`	Dołącza do łańcucha zapisanego w pierwszym argumencie wywołania łańcuch z drugiego argumentu.
`strncat(str1,str2,n);`	Jak wyżej, ale ogranicza się do `n` znaków drugiego łańcucha.

Funkcje wykonujące inne operacje na łańcuchach znaków dostępne są po włączeniu do programu pliku nagłówkowego `strings.h` lub `string.h`. Tabela 1 zawiera wykaz kilku z najczęściej używanych. Używając funkcji, które modyfikują zawartość zmiennych (np. dodają do ciągu nowe znaki lub kopiują ciągi znaków) należy pamiętać, że **nie sprawdzają one pojemności zmiennych do których zmodyfikowane ciągi są zapisywane**. Trzeba być ostrożnym, aby nie zapisywać danych poza zmienną.

2.2. Przykłady innych funkcji operujących na ciągach

Ten rozdział zawiera opisy funkcji, które nie wymienionych w tabeli 1, ale realizujących równie ważne operacje na łańcuchach znaków.

2.2.1. Funkcja atoi()

```c
#include <stdlib.h>
int atoi(const char *nptr);
```

Listing 4: Prototyp funkcji `atoi()`

Funkcja `atoi()` konwertuje pierwszą liczbę zapisaną w ciągu na wartość typu `int`.

Tabela 2: Parametry funkcji `atoi()`

Parametr	Opis
`nptr`	Ciąg zawierający liczbę do konwersji.
`Wartość zwracana`	Liczba odczytana z ciągu i podana jako wartość typu `int`.

2.2.2. Funkcja strstr()

```c
#include <string.h>
char *strstr(const char *haystack, const char *needle);
```

Listing 5: Prototyp funkcji `strstr()`

Funkcja `strstr()` znajduje pierwsze wystąpienie podciągu `needle` w ciągu `haystack`. Zerowe bajty kończące ciągi ('\0') nie są porównywane.

Tabela 3: Parametry funkcji `strstr()`

Parametr	Opis
`haystack`	Ciąg, w którym wyszukiwany jest podciąg.
`needle`	Podciąg, który jest poszukiwany.
`Wartość zwracana`	Funkcja zwraca wskaźnik do miejsca, gdzie znajduje się pierwsza litera pierwszego wystąpienia szukanego podciągu. Jeżeli podciąg nie zostanie znaleziony, wówczas zwracana jest wartość `NULL`.

2.2.3. Funkcja strtok()

Funkcja `strtok()` zamienia pojedynczy ciąg na sekwencję niepustych podciągów, zwanych tokenami. W pierwszym parametrze funkcji podajemy ciąg do podziału. Jeżeli chcemy uzyskać kolejny token z tego samego ciągu, w parametrze tym musimy podać wartość `NULL`. Drugim parametrem jest lista znaków,

```c
#include <string.h>
 char *strtok(char *str, const char *delim);
```

Listing 6: Prototyp funkcji `strtok()`

które dzielą pierwszy ciąg na tokeny.

UWAGA! Funkcja używa zmiennej statycznej, dlatego nie jest bezpieczna, jeżeli jest ona używana w różnych częściach programu równocześnie. W takiej sytuacji wynik działania programu może być niezgodny z oczekiwaniami.

Tabela 4: Parametry funkcji `strtok()`

Parametr	**Opis**
`str`	Ciąg do podziału na tokeny. Podajemy tylko przy pierwszym wywołaniu funkcji dla danego ciągu. Jeżeli chcemy uzyskać kolejny token z tego samego ciągu, przez ten parametr należy przekazać wartość `NULL`.
`delim`	Tablica znaków dzielących ciąg z parametru pierwszego na tokeny.
`Wartość zwracana`	Zwraca wskaźnik na następny token lub `NULL` jeżeli nie ma więcej tokenów.

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

int main(void)
{
    char ciag[] = "Tekst do podzialu na slowa (nie znaki). Np: Ala ma - kota i psa";
    char znaki_podzialu[] = " .-:()";
    char * token;
    printf( "Podzial tekstu \"%s\" na pojedyncze wyrazy:\n", ciag );
    token = strtok( ciag, znaki_podzialu );
    while( token != NULL ) {
        printf( "%s\n", token );
        token = strtok( NULL, znaki_podzialu );
    }
    return 0;
}
```

Listing 7: Przykład użycia funkcji `strtok()`

Wynik działania programu:

```
Podzial tekstu "Tekst do podzialu na slowa (nie znaki). Np: Ala ma - kota i psa" na pojedyncze
wyrazy:
Tekst
do
podzialu
na
slowa
nie
znaki
Np
Ala
ma
kota
i
psa
```

3. Zadania

Wszystkie programy należy napisać z podziałem na funkcje z parametrami!

1. Wykonaj w trybie krokowym debuggera programu z listingu 7. Sprawdź jak zmienia się zawartość zmiennej `ciag`.
2. Napisz program, który sprawdzi, czy podane przez użytkownika słowo jest palindromem.
3. Napisz program, który sprawdzi, czy podane przez użytkownika dwa słowa są anagramami.
4. Napisz program, który pozwoli wprowadzić użytkownikowi liczbę rzeczywistą z przecinkiem zamiast kropki, zwiększy jej wartość o 0,001, a następnie wypisze wynik na ekranie również z przecinkiem. Dokonaj odpowiednich konwersji w osobnych funkcjach.
5. Napisz program, który odwróci kolejność wyrazów w zdaniu wprowadzonym przez użytkownika.
6. Napisz program, który znajdzie wszystkie wystąpienia określonego wzorca w zadanym ciągu i zastąpi je innym ciągiem. Wzorzec, ciąg wstawiany oraz ciąg przeszukiwany będą podawane przez użytkownika.
7. Napisz program, który pobierze od użytkownika adres URL, wypisze go na ekran, a w osobnym wierszu poda identyfikator protokołu z tego adresu, o ile został on w nim podany.