Podstawy programowania
(wykład)

Łańcuchy znakowe

dr inż. Robert Arsoba
Robert.Arsoba@tu.koszalin.pl

Koszalin 2022
Wersja 1.0

Łańcuchy znakowe

Łańcuchy znakowe

Łańcuch znakowy w języku C

W języku C nie istnieje typ danych przeznaczony dla łańcuchów znakowych (napisów). Stosowane są w tym celu dwa rozwiązania:

- **tablica znakowa** (długość ograniczona rozmiarem tablicy)
- **wskaźnik `char*`** (tzw. **C-string**, dowolna długość)

Łańcuch znakowy jest tablicą znaków zakończoną znakiem pustym (znak o kodzie 0, znak `'\0'`, stała **`NULL`**).

```
char napis1[10];
/* łańcuch o długości 10 znaków */

char *napis2;
/* łańcuch o nieokreślonej długości */
/* koniec łańcucha w pamięci to znak o kodzie 0 */
```

Łańcuchy znakowe

Inicjalizacja łańcuchów znakowych

```
char napis1[7] = { 'A', 'N', 'S', 'I', ' ', 'C', '\0' };

char napis2[7] = "ANSI C";
```

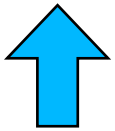
Na końcu należy samodzielnie dodać znak `'\0'`.

Na końcu automatycznie dodawany jest znak `'\0'`.

W obu przypadkach napisy jako tablice są modyfikowalne, a rozmiar tablicy można pominąć [ ] (zostanie obliczony przez kompilator).

A	N	S	I		C	'\0'

```
char *napis3 = "ANSI C";
```



Wskaźnik **`napis3`** pokazuje adres pamięci pod którym przechowywany jest pierwszy znak łańcucha. Łańcuch kończy się w pamięci tam, gdzie znajduje się znak `'\0'`.

Ważne: zainicjalizowany w ten sposób napis **`char*`** jest **niemodyfikowalny** (tylko do odczytu).

Łańcuchy znakowe

Umieszczanie łańcucha znakowego w zmiennej

W tablicy znakowej napis umieszczamy funkcją **strcpy**

```
char napis1[6];          /* maks. 5 znaków + '\0' */
strcpy(napis1, "Hello");       /*  kopiowanie  */
```

Zwykłe przypisanie `napis1 = "Hello";` jest nieprawidłowe!

W przypadku wskaźnika, napis umieszczamy poprzez przypisanie

```
char *napis2;            /* dowolnie długi napis */
/* tu konieczna alokacja pamięci dla napisu  */
napis2 = "ANSI C";    /* podstawienie              */

gets(napis1);       /* wczytanie łańcucha z stdin */
fgets(napis1, 6, stdin);    /* wczytanie z stdin */
```

Funkcja **gets** przerywa wczytywanie łańcucha po znaku końca wiersza (**ENTER**), nie przechowuje go i dołącza na końcu łańcucha znak `'\0'`. Pozwala wczytać łańcuchy zawierające spacje i tabulatory. Niedostępna od standardu C11 (zalecane użycie **fgets**). Funkcja **fgets** przechowuje znak końca wiersza i dołącza na końcu znak `'\0'`. Drugi argument **n** (maks. liczba znaków) uwzględnia znak końca wiersza i znak `'\0'`. Funkcja **fgets** przerywa wczytywanie po znaku końca wiersza (**ENTER**) lub po wczytaniu **n-1** znaków.

Łańcuchy znakowe

Wczytywania łańcucha znakowego z klawiatury

```
char bufor[6];
```

Brak możliwości wczytywania napisów zawierających spacje

funkcja	ryzyko przepełnienia bufora	spacja kończy odczyt	uwagi
`scanf("%s", bufor);`	tak	tak	nazwa tablicy to adres pierwszego elementu (brak &)
`scanf("%5s", bufor);`	nie	tak	
`gets(bufor);`	tak	nie	niedostępne od C11, nie jest przechowywany znak końca wiersza
`fgets(bufor, 6, stdin);`	nie	nie	przechowywany jest znak końca wiersza (klawisz **ENTER**)

Przykład eliminacji znaku końca wiersza:

```
fgets(bufor, 6, stdin);
bufor[strlen(bufor)-1] = '\0';
```

Łańcuchy znakowe

Przykłady przetwarzania łańcuchów znakowych

Funkcja jako wynik zwraca długość napisu **`s`**. Odpowiednik **`strlen`**.

Wersja 1

```
int dlugosc(const char *s)
{
   int i = 0;
   while (*s++ != '\0')
      i++;
   return i;
}
```

Wskaźnik na stałą typu **`char`** czyli stały napis (napis **`s`** jest niemodyfikowalny, tylko do odczytu)

Wersja 2

```
int dlugosc(const char *s)
{
   int i;
   for (i = 0; *s++ != '\0'; i++)
      ;              /*  continue;  */
   return i;
}
```

Łańcuchy znakowe

Przykłady przetwarzania łańcuchów znakowych

Funkcja jako wynik zwraca długość napisu **`s`**. Odpowiednik **`strlen`**.

Wersja 3

```
int dlugosc(const char *s)
{
    const char *p = s;

    while (*p++ != '\0')
        ;

    return p-s-1;
}
```

Wykorzystana arytmetyka na wskaźnikach (różnica wskaźników):
`p` wskazuje koniec napisu
`s` wskazuje początek napisu
`-1` czyli długość bez znaku **`'\0'`**

Łańcuchy znakowe

Przykłady przetwarzania łańcuchów znakowych

Funkcja kopiuje łańcuch **`src`** do tablicy wskazywanej przez **`dest`**.

Odpowiednik funkcji bibliotecznej **`strcpy`**.

Wersja 1

```
char* kopiuj(char *dest, const char *src)
{
    char *p = dest;

    while (*src != '\0')
        *p++ = *src++;

    *p = '\0';
    return dest;
}
```

Znak końca napisu nie jest kopiowany

Zaznaczenie końca napisu

Zwracany jest adres początku napisu

Łańcuchy znakowe

Przykłady przetwarzania łańcuchów znakowych

Funkcja kopiuje łańcuch **`src`** do tablicy wskazywanej przez **`dest`**.

Odpowiednik funkcji bibliotecznej **`strcpy`**.

Wersja 2

```
char* kopiuj(char *dest, const char *src)
{
    char *p = dest;

    /*   while (*p++ = *src++);   */
    while ( (*p++ = *src++) != '\0' )
        ;

    return dest;
}
```

„skrót” myślowy

Znak końca napisu jest kopiowany

Zwracany jest adres początku napisu

Łańcuchy znakowe

Funkcje do przetwarzania łańcuchów znakowych (**`string.h`**)

`strcat, strncat` - łączy łańcuchy znakowe

`strchr` - sprawdza, czy łańcuch znakowy zawiera podany znak

`strcmp, strncmp` - porównuje łańcuchy znakowe

`strcpy, strncpy` - kopiuje łańcuch do tablicy (łącznie z **`'\0'`**)

`strlen` - zwraca długość łańcucha (nie wliczając znaku **`'\0'`**)

`strstr` - wyszukuje podłańcuch w łańcuchu znakowym

`strtok` - dzieli łańcuch na tzw. tokeny (słowa, jednostki leksykalne)

`strlwr` - *zamienia w łańcuchu znakowym duże litery na małe*

`strupr` - *zamienia w łańcuchu znakowym małe litery na duże*

`strrev` - *ustawia znaki łańcucha w odwrotnej kolejności (wspak)*

} spoza ANSI C

Szczegółowy opis funkcji: www.cplusplus.com/reference/cstring

Łańcuchy znakowe

Wybrane funkcje do konwersji łańcuchów znakowych (**`stdlib.h`**)

`atoi` - zamienia łańcuch na liczbę typu **`int`**

`atol` - zamienia łańcuch na liczbę typu **`long`**

} łańcuch musi zawierać liczbę całkowitą

`atoll` - zamienia łańcuch na liczbę typu **`long long`** — jak wyżej, dostępne od C99

`atof` - zamienia łańcuch na liczbę typu **`double`**

łańcuch musi zawierać liczbę zmiennoprzecinkową

`strtol` - zamienia łańcuch na liczbę typu **`long`**

`strtoul` - zamienia łańcuch na liczbę typu **`unsigned long`**

} łańcuch może zawierać liczbę całkowitą w systemie liczbowym o dowolnej podstawie 2÷36

`strtod` - zamienia łańcuch na liczbę typu **`double`** (dodatkowy arg.)

`strtof` - zamienia łańcuch na liczbę typu **`float`** (dostępne od C99)

Szczegółowy opis funkcji: www.cplusplus.com/reference/cstdlib

Łańcuchy znakowe

Funkcje do przetwarzania znaków (`ctype.h`)

argument typu **`int`** (kod znaku to najmłodszy bajt)

`isalnum` - sprawdza, czy znak jest literą lub cyfrą

`isalpha` - sprawdza, czy znak jest literą

`isdigit` - sprawdza, czy znak jest cyfrą

`islower` - sprawdza, czy znak jest małą literą

`isupper` - sprawdza, czy znak jest dużą literą

`iscntrl` - sprawdza, czy znak jest znakiem sterującym (0÷31, 127)

`isprint` - sprawdza, czy znak jest „drukowalny”, niesterujący (32÷126)

`ispunct` - sprawdza, czy znak jest znakiem interpunkcyjnym

`isspace` - sprawdza, czy znak to znak biały, separujący (9÷13, 32)

(**`isblank`** - dostępne od C99, sprawdza, czy znak to spacja lub tabulator)

`isxdigit` - sprawdza, czy znak jest cyfrą szesnastkową 0÷9, A÷F, a÷f

`tolower` - dla argumentu, który jest dużą literą, zwraca małą literę

`toupper` - dla argumentu, który jest małą literą, zwraca dużą literę

Szczegółowy opis funkcji: www.cplusplus.com/reference/cctype