Ćwiczenie nr 6

Temat: Operacje na łańcuchach znaków.

Zagadnienia:
- Zasady pracy z łańcuchami tekstowymi (tablice wartości typu char).
  - funkcje standardowe operacji na łańcuchach,
  - funkcje I/O dla operacji na łańcuchach znaków.

1. Przechowywanie danych tekstowych

W standardzie języka ANSI C nie ma specjalnego typu danych umożliwiającego przechowanie i operowanie na tekstach. Standardowo łańcuchy znakowe (strings) przechowuje się w tablicach elementów typu char, np.:

```
char tekst[100];
```

Poprawne deklaracje takich zmiennych „tekstowych" mogą wyglądać tak:

```
char *tekst ="Jan Maria";
char tekst[] ="Jan Maria";
char tekst[] = {'J','a','n',' ','M','a','r','i','a','\0'} ;
```

deklaracje te są równoważne. Także taka deklaracja:

```
char tekst[100] = "Jan Maria";
```

też jest poprawna ale rezerwowany jest stały obszar pamięci (100 bajtów) i w dalszej części programu można wykorzystać 100-bajtową tablicę.

W standardowych bibliotekach ANSI C mamy dostęp do kilkunastu funkcji (patrz pkt. 3 – Lista funkcji operujących na łańcuchach znaków) operujących na takich właśnie tablicach. W operacjach na tablicach znakowych obowiązuje zasada: **Tekst w tablicy znaków typu char kończy się wartością zero (ang. *null-terminated*)**. Inaczej „mówiąc" - funkcje dostępne w standardowych bibliotekach traktują wartość zero w tablicy jako koniec tekstu. Nie należy mylić z umieszczeniem znaku '0' w tablicy... Koniec tekstu jest to wartość zero a nie kod znaku '0'! Można zakończyć tekst w łańcuchu tekstowym np. tak:

```
char tekst[100] = "Nie chce mi się dzisiaj programowac";
tekst[3] = 0; /* zero konczy tekst dla funkcji */
   lub
tekst[3] = '\0'; /* to jest to samo */
   ale
tekst[3] = '0'; /* to jest umieszczenie w tablicy kodu znaku 0 i to
                   nie jest zakonczenie tekstu! */
```

2. Operacje wejścia/wyjścia (I/O)

Pobranie danych do tablicy znakowej (łańcucha znaków) można wykonać m.in. za pomocą funkcji scanf:

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

int main(void)
{
  char name[20];

  printf("Podaj imie: ");
  scanf("%20s", name); /* %20s - pobrane będzie nie więcej niż 20 znakow */

  printf("%s \n", name);

  system("pause");
  return 0;
}
```

ale jeżeli podamy ciąg znaków z tzw. białym znakiem (ze spacją, tabulacją itd.) np. „Jan Kowalski” w zmienej *name* zostanie umieszczone tylko pierwsze słowo tzn. „Jan”. Jeżeli chcemy pobrać cały tekst wraz ze spacjami można to wykonać za pomocą funkcji **gets**:

```
#include <stdio.h>

int main(void)
{
  char name[128];
  printf("Podaj imie: ");

  gets(name); /* pobranie danych do tablicy name */

  printf("%s \n", name); /* drukowanie danych */
  system("pause");
  return 0;
}
```

3. Zadania

zad. 1. Program wyznaczający na podstawie podanego przez użytkownika numeru pesel (np. 84052356789) wydrukuje informację o dacie urodzenia. Podstawowy format kodowania danych w PESEL (dla osób urodzonych w XX wieku):

rok urodzenia (pierwsze dwie cyfry), miesiąc urodzenia (cyfry 3-4), dzień miesiąca urodzenia (cyfry 5-6).

zad. 2. Program, który po wprowadzeniu z konsoli słowa np. "KAWA" wyświetli na konsoli tekstowej:

```
AAAA
WWW
AA
K
K
AA
WWW
AAAA
```

zad. 3. Program przeprowadzający krótką ankietę z użytkownikiem, np:

```
Jak masz na imię?
Hans
Podaj Twoje nazwisko:
Klinkenhofen
Ile masz lat?
12
```

plus kilka innych pytań wg uznania. Na koniec program wyświetli podsumowanie, np: Nazywasz się Hans Klinkenhofen i masz 12 lat.

Dodatek. Lista funkcji operujących na łańcuchach znaków (string.h)

Zestawienie standardowych funkcji operacji na łańcuchach tekstowych, biblioteka string - plik nagłówkowy *string.h*:

char strcpy (char *s, const char *ct)
Kopiuje zawartość tablicy znaków ct do tablicy s łącznie ze znakiem '\0'; zwraca s. Funkcja nie sprawdza długości tablicy, do której odbywa się kopiowanie, stąd tablica docelowa musi mieć miejsce na co najmniej n+ l znaków, gdzie n to ilość znaków w tablicy ct.

char strncpy (char *s, const char *ct, size_t n)
Kopiuje znaki z tablicy znaków ct do tablicy s, jednak nie więcej niż n znaków. Jeżeli w ct jest mniej niż n znaków, puste miejsce dopełniane jest znakami \0. Funkcja zwraca s.

char strcat (char *s, const char *ct)
Dopisuje znaki z tablicy znaków ct na koniec tablicy s, zwracane jest s.

char strncat(char *s, const char *ct, size_t n)
Dołącza co najwyżej n znaków z tablicy ct do końca tablicy s, tablica s kończy się znakiem '\0'. Funkcja zwraca s.

char strcmp (const char *cs, const char *ct)
Porównuje tablice znaków cs i ct. Zwracana jest: wartość <0 dla cs < ct wartość 0 dla cs jest równego ct wartość >0 dla cs > ct. Porównywanie rozpoczyna się od pierwszego znaku obu tablic do momentu, gdy znaki będą się różnić lub do osiągnięcia końca jednej z tablic. Kryterium porównania jest wartość kodu ASCII danego znaku.

char strncmp (const ch ar *cs, const char *ct)
Funkcja ma podobne działanie jak funkcja strcmp z tą różnicą, że porównywanych jest co najwyżej n znaków.

char *strchr (const char *cs, int c)

Zwraca wskaźnik do pierwszego wystąpienia znaku c w cs. w wypadku nie wystąpienia tego znaku zwracane jest NULL.

char *strrchr (const char *cs, int c)
Zwraca wskaźnik do ostatniego wystąpienia znaku c w tablicy cs. Jeśli znak nie występuje, zwracane jest NULL.

size_t strspn (const char *cs, const ch ar *ct)
Zwraca długość przedrostka w tablicy cs składającego się ze znaków występujących w tablicy ct.

size_t strcspn (const char *cs, const char *ct)
Działa podobnie jak funkcja strspn z tym, że zwraca długość przedrostka w cs składającego się ze znaków nie występujących w ct, funkcja przerywa zliczanie, gdy w cs napotka znak występujący w ct.

char *strpbrk (const char *cs, const char *ct)
Zwraca wskaźnik do pierwszego wystąpienia w tablicy cs któregokolwiek ze znaków z tablicy ct. W przypadku nie znalezienia żadnego znaku zwraca NULL.

char *strstr (const char *cs, const char *ct)
Zwraca wskaźnik do pierwszego wystąpienia cs w tablicy ct lub wartość NULL.

size_t strlen (const char *cs)
Zwraca długość tablicy cs bez uwzględnienia końcowego znaku '\0'.

char *strerror (size_t n)
Zwraca wskaźnik do tekstu komunikatu odpowiadającemu błędowi o numerze n. Wywołana bez numeru wypisuje komunikat o ostatnim napotkanym błędzie.

char *strtok (char *s, const char *ct)
Wyszukuje w tablicy s ciągi znaków przedzielone znakami z tablicy ct. Kolejne wywołania tej funkcji dzielą tablicę s na ciągi znaków rozdzielone znakami z tablicy ct. Funkcja po wywołaniu znajduje pierwszy ciąg znaków nie należących do ct. Znak następny zastępowany jest znakiem /0 i zwracany jest wskaźnik do początku tego ciągu. Każde następne wywołanie tej funkcji musi być z argumentem s równym NULL, wówczas zwracany jest kolejny ciąg znaków, przy czym szukanie rozpoczyna się za końcem poprzedniego ciągu. Zwraca NULL, gdy nie znajduje ciągów. Każde następne wywołanie może być z innym argumentem ct.

char *strlwr(char *s);
Zamienia w tablicy s litery duże na małe. Pozostałe znaki nie są zmieniane. Zwraca wskaźnik do s.

char *strupr(char *s);
Zamienia w tablicy s litery małe na duże. Pozostałe znaki nie są zmieniane. Zwraca wskaźnik do s.

Funkcje operacji na pamięci, przydatne przy pracy na łańcuchach, również plik nagłówkowy *string.h*:

void *memcpy (void *s, const void *ct, size_t n)
Kopiuje n znaków z obiektu ct do obiektu s i zwraca s.

void *memmove (void *s, const void *ct, size_t n)
Wstawia n znaków z obiektu ct do obiektu s i zwraca s. Różni się od memcpy tym, że pozostawia w obiekcie s jego końcową część, jeżeli obiekt s ma większą długość niż obiekt ct.

int memcmp (void const *cs, const void *ct, size_t n)
Porównuje początkowe n znaków zawartych w obiektach ct i cs. Zasada porównywania jest taka sama jak dla funkcji strcmp. Zwracana jest: wartość <0 dla cs < ct wartość 0 dla cs równego ct wartość >0 dla cs > ct.

void *memchr (void const *cs, int c, size_t n)
Zwraca wskaźnik do pierwszego wystąpienia znaku c w obiekcie cs lub wartość NULL, gdy brak znaku c. Sprawdza tylko n początkowych znaków.

void *memset (void *s, int c, size_t n)
Wstawia znak c do początkowych n znaków obiektu s. Zwracane jest s.