Lista zadań nr 2

Łańcuchy znaków (napisy).

Zadania podstawowe:

Zadanie 1 Napisz i wypróbuj własną wersję funkcji `strcat()`. Spróbuj napisać swoją definicję najkrócej jak tylko to możliwe.

Zadanie 2 Zaprojektuj i sprawdź funkcję wyszukującą w łańcuchu przekazanym w pierwszym argumencie pierwszego wystąpienia znaku podanego w drugim argumencie. W przypadku znalezienia znaku, funkcja ma zwrócić wskaźnik do niego, w przeciwnym wypadku wskaźnik pusty (w podobny sposób działa funkcja `strchr()`).

Zadanie 3 Napisz i przetestuj funkcję, która odwraca zawartość łańcucha i zapisuje go w tym samym miejscu.

Zadanie 4 Napisz i przetestuj funkcję, która pobiera łańcuch i usuwa z niego odstępy. Funkcja nie powinna wykorzystywać dodatkowego bloku pamięci do wykonania tej operacji.

Zadanie 5 Zaprojektuj i przetestuj funkcję, która pobiera ze standardowego wejścia pierwsze słowo i porzuca resztę wiersza. Za słowo można użyć ciąg znaków niezawierający spacji, tabulatorów lub znaków nowej linii.

Zadanie 6 Uzupełnij poniższą funkcję, tak by wyświetlała pierwsze trzy znaki napisu umieszczonego w tablicy o elementach typu `char` będącego argumentem funkcji. Uwzględnić napisy, które zawierają mniej niż trzy znaki.

```
void print_three(char *str)
{
    ...
}
```

Zadanie 7 Napisz funkcję, która używa funkcji `strcmp()` do porównania dwóch ciągów tekstowych przekazanych jako argumenty. Funkcja powinna wyświetlać informację o tym, czy pierwszy ciąg tekstowy jest mniejszy, równy czy większy niż drugi ciąg tekstowy. Przetestuj funkcję w programie, który pobiera pary ciągów tekstowych od użytkownika.

Zadanie 8 Napisz funkcję, która używa funkcji `strncmp()` do porównania dwóch ciągów tekstowych przekazanych jako argumenty. Trzecim argumentem powinna być liczba porównywanych znaków. Funkcja powinna wyświetlać informację o tym, czy pierwszy ciąg tekstowy jest mniejszy, równy czy większy niż drugi ciąg tekstowy. Przetestuj funkcję w programie, który pobiera pary ciągów tekstowych od użytkownika oraz liczbę porównywanych znaków.

Zadanie 9 Utwórz program wykorzystujący generator liczb pseudolosowych do utworzenia zdań. Program powinien używać czterech tablic (`article`, `noun`, `verb` i `preposition`) wskaźników do `char`. Działanie programu ma polegać na utworzeniu zdania przez losowy wybór słowa z każdej tablicy w podanej kolejności: `article`, `noun`, `verb`, `preposition`, `article` i `noun`. Po pobraniu każdego słowa ma zostać ono połączone z poprzednimi w tablicy wystarczająco dużej do przechowywania całego zdania. Poszczególne słowa mają być rozdzielone spacjami. Program powinien wygenerować 20 takich zdań, w tablicy `article` mogą być umieszczone elementy takie jak `"ten"`, `"ta"`, `"to"` itd. W tablicy `noun` mogą być umieszczone elementy takie jak `"chłopak"`, `"dziewczyna"`, `"pies"`, `"miasto"`, `"samochod"` itd. W tablicy `noun` mogą być umieszczone elementy takie takie jak `"prowadzil"`, `"skoczyl"`, `"uciekl"`, `"szedl"`, `"przeskoczyl"` itd. Z kolei w tablicy `preposition` mogą być umieszczone elementy takie jak `"do"`, `"z"`, `"na"`, `"nad"`, `"pod"` itd. Gdy utworzysz program i upewnisz się o jego działaniu, zmodyfikuj go w taki sposób, aby generował krótkie historie na podstawie kilku takich zdań.

Zadanie 10 Utwórz program konwertujący wyrażenia w języku angielskim na tzw. świńską łacinę, czyli zakodowaną konstrukcję językową często używaną w celach rozrywkowych. Istnieje wiele rożnych metod tworzenia wyrażeń w świńskiej łacinie. Dla uproszczenia wykorzystaj przedstawiony tutaj algorytm.

Aby wyrażenie w świńskiej łacinie utworzyć na podstawie istniejącego wyrażenia w języku angielskim, należy najpierw to wyrażanie stokenizować na słowa za pomocą funkcji `strtok()`. Konwersja słowa angielskiego na słowo w świńskiej łacinie odbywa się następująco: pierwszą literę przenieś na koniec słowa, a następnie dodaj litery `ay`. W ten sposób `jump` zmienia się w `umpjay`, słowo `the` zamienia się na `hetay`, a słowo `computer` na zmienia się `oputercay`. Spacje pomiędzy słowami pozostają bez zmian. Przyjmij również następujące założenia:

- wyrażenie w języku angielskim składa się ze słów oddzielonych spacjami;
- w wyrażeniach nie są używane znaki przestankowe;
- każde słowo składa się przynajmniej z dwóch liter.

Funkcja `print_latin_word()` powinna wyświetlać poszczególne słowa - po znalezieniu tokenu przez wywołanie `strtok()` prowadzący do niego wskaźnik powinien zostać przekazany funkcji `print_latin_word()`, a następnie ma być wyświetlone słowo w świńskiej łacinie.

Zadanie 11 Utwórz program pobierający ciąg tekstowy numeru telefonu w postaci `(555) 555-5555`. Program powinien używać funkcji `strtok()` do wyodrębniania tokenów w postaci numeru kierunkowego, pierwszych trzech cyfr numeru telefonu i ostatnich czterech cyfr

numeru telefonu. Następnie siedem cyfr tworzących numer telefonu ma zostać połączonych w jeden ciąg tekstowy. Program powinien skonwertować na wartość typu `int` ciąg tekstowy numeru kierunkowego oraz na wartość typu `long int` ciąg tekstowy numeru telefonu. W wyniku działania programu mają zostać wyświetlone numer kierunkowy i numer telefonu.

Zadanie 12 Utwórz program, który pobiera wiersz tekstu, tokenizuje go za pomocą funkcji `strtok()`, a następnie wyświetla tokeny w odwrotnej kolejności.

Zadanie 13 Utwórz program, który pobiera od użytkownika tekst oraz pewną frazę. Wykorzystując funkcję `strstr()`, program powinien wyszukać pierwsze wystąpienie tej frazy w podanym wierszu tekstu, a następnie znalezione położenie przypisać do zmiennej `search_ptr` typu `char*`. Jeżeli szukana fraza zostanie znaleziona, należy wyświetlić pozostałą część wiersza, począwszy od znalezionej frazy. Następnie funkcja `strstr()` ma zostać użyta do znalezienia kolejnego wystąpienia frazy w wierszu tekstu. Jeżeli zostanie znalezione drugie wystąpienie szukanej frazy , należy wyświetlić pozostałą cześć tekstu począwszy od znalezionego drugiego wystąpienia szukanej frazy. Wskazówka: drugie wywołanie funkcji `strstr()` powinno zawierać pierwszy argument w postaci `search_ptr + 1`.