Lista zadań nr 3

Łańcuchy znaków, pliki tekstowe.

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 ,KMJnm,JKJK 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 i 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`.

Zadanie 14 Na podstawie programu opracowanego w zadaniu 13 napisz program, który przetwarza plik tekstowy wiersz po wierszu i zliczy całkowitą liczbę wystąpień danej frazy w tym tekście. Program powinien wyświetlić wynik tej operacji. Na potrzeby programu napisz funkcję, która dla danego ciągu tekstowego i przekazanej frazy zwróci liczbę wystąpień tej frazy w podanym ciągu tekstowym.

Zadanie 15 Utwórz program analogiczny do tego z zadania 14, ale zliczający liczbę wystąpień podanego znaku w danym pliku tekstowym - wykorzystaj funkcję `strchr()`.

Zadanie 16 Na podstawie programu z zadania napisz program, który ustali całkowitą liczbę wystąpień poszczególnych liter alfabetu w danym pliku tekstowym. Małe i wielkie litery mają być zliczane razem. Wartości określające liczbę wystąpień poszczególnych liter mają być przechowywane w tablicy. Po zliczeniu wszystkich wystąpień program powinien wyświetlić w formacie tabelarycznym wynik tej operacji. Na potrzeby programu napisz funkcję, która dla przekazanego jej ciągu tekstowego i przekazanej tablicy zliczy wystąpienia poszczególnych liter alfabetu w podanym ciągu i wyniki zapisze w przekazanej tablicy.

Zadanie 17 Utwórz program przetwarzający plik tekstowy wiersz po wierszu, który zliczy liczbę słów w tym pliku - wykorzystaj funkcję `strtok()`. Należy przyjąć założenie, że są rozdzielone spacjami lub znakami nowego wiersza.

Zadanie 18 Utwórz plik tekstowy zawierający kilkadziesiąt miejscowości z twojej okolicy zapisanych w osobnych wierszach. Zlicz liczbę wierszy i wczytaj słowa do tablicy łańcuchów

stworzonej dynamicznie. Wiersze tablicy powinny być tylko tak długie jak długie są poszczególne słowa (wraz ze znakiem `'\0'`). W tym celu wczytuj słowa do tablicy tymczasowej (dostatecznie dużej) i kopiuj je do dynamicznie tworzonej tablicy (wiersza tablicy łańcuchów) - wykorzystaj funkcję `strlen()` i `strcpy()`. Wykorzystując funkcje przeznaczone do porównywania ciągów tekstowych i znane Ci techniki sortowania, posortuj tak utworzoną tablicę łańcuchów. Na potrzeby programu stwórz funkcje:

- zliczającą liczbę wierszy w piku (wskaźnik fdo `FILE`) przekazanym jej jako argument - funkcja powinna zwracać tę liczbę;
- funkcję tworzącą dynamicznie tablicę łańcuchów, wypełniającą ją ciągami pobranymi z pliku;
- funkcję sortująca tablicę łańcuchów;
- funkcję wyświetlającą tablicę łańcuchów.

Zadanie 19 Napisać program, który dzieli zawartość wskazanego przez użytkownika pliku na dwie części i zapisuje każdą z tych części do osobnego pliku. Użytkownik powinien podać nazwę pliku i rozmiar pierwszej części. Jak zmodyfikować ten program, by zapisywał zawartość wskazanego pliku do plików, których rozmiar nie przekracza podanej wartości (liczba tworzonych plików zależy od rozmiaru "dzielonego" pliku).

Zadanie 20 Uzupełnić funkcję `read_word()`:

```
void read_word(char *str, int n, FILE *fp)
{
    ...
}
```

Wywołując funkcję `read_word()` należy jako pierwszy argument podać tablicę, w której umieszczany jest odczytywany wyraz. Maksymalna liczba umieszczanych w tej tablicy znaków podawana jest za pomocą drugiego argumentu (nie więcej niż $n - 1$ znaków odczytanych z pliku, a po nich zero). Trzeci argument wskazuje plik, z którego mają być odczytywane znaki. Jeśli funkcja została wywołana w podany poniżej sposób:

```
int main()
{
 char t[8];
 FILE *plik = fopen("byleco.txt","r");
 if(plik == NULL){
     printf("plik␣nie␣zostal␣otwarty");
```

```
        return 1;
    }

    read_word(t, 8, plik);
    printf(t, %s\n);
    read_word(t, 8, plik);
    printf(t, %s\n);
    fclose(plik);
    return 0;
}
```

a zawartość pliku *byleco.txt* jest następująca:

```
( wyraz ) , innywyraz oraz
```

to pierwsze wywołanie funkcji `read_word()` powinno pominąć dwa pierwsze znaki nie będące literami (odstęp i nawias) i umieścić w tablicy pięć kolejnych liter odczytanych z pliku (wyraz) oraz 0. Funkcja `fgetc()` pozwala czytać po jednym znaku z pliku, a funkcja `isalpha()` sprawdzić, czy odczytany znak jest literą. Aby znaleźć koniec wyrazu funkcja musi też odczytać znajdujący się po nim nawias. Jeśli jest to potrzebne, znak ten można "wycofać" do pliku funkcją `ungetc()`.

Tablica `t` po pierwszym wywołaniu funkcji `read_word()`:

'w'	'y'	'r'	'a'	'z'	'\0'		

Kolejne wywołanie funkcji `read_word()` wstawia do tablicy `t` siedem pierwszych znaków (`8-1`) wyrazu "innywyraz" i umieszcza po nich zero. Kolejne dwa znaki ("a" oraz "z") też są odczytywane, ale funkcja "nic z nimi nie robi". Funkcja kończy swoje działanie po odczytaniu znaku odstępu znajdującego się przed wyrazem "oraz" (gdyby funkcja `read_word()` została wywołana jeszcze raz, to powinna odczytać wyraz "oraz").

Tablica `t` po drugim wywołaniu funkcji `read_word()`:

'i'	'n'	'n'	'y'	'w'	'y'	'r'	'\0'

Zadanie 21 W pliku *liczby.txt* znajduje się 1000 liczb naturalnych zapisanych binarnie. Każda liczba zapisana jest w osobnym wierszu. Pierwsze pięć wierszy zawiera następujące liczby:

```
11001100010000111001
11101000111011001
1111010011100011101100
101001110111
10101111100111111000000010111111011
```

Każda liczba binarna zawiera co najwyżej 64 cyfr binarnych. Napisz program, który zapisuje podane liczby jako ciągi tekstowe w tablicy alokowanej dynamicznie typu `char*` o rozmiarze

1000, gdzie pamięć dla poszczególnego ciągu również powinna być alokowana dynamicznie (przydzielona pamięć powinna być dokładnie równa długości danego ciągu cyfr binarnych - pamiętnej o miejscu na znak '\0'). Opisane zadanie tworzenia tablicy i wypełniania jej odpowiednimi danymi z pliku powinno być realizowane przez oddzielną funkcję zwracającą wskaźnik do tej tablicy.

Napisz program, który da odpowiedzi do poniższych podpunktów - do wykonania każdego z podpunktów napisz osobną funkcję. Odpowiedzi zapisz w pliku *wyniki1.txt*, a każdą odpowiedź poprzedź numerem oznaczającym odpowiedni podpunkt.

a) podaj, ile liczb z pliku *liczby.txt* ma w swoim zapisie binarnym więcej zer niż jedynek;

b) podaj, ile liczb w pliku *liczby.txt* jest podzielnych przez 2 oraz ile liczb jest podzielnych przez 8;

c) znajdź najmniejszą i największą liczbę w pliku *liczby.txt* - jako odpowiedź podaj numery wierszy, w których się one znajdują.

Zadanie 22 W ramach projektu ALPHA naukowcom udało się odczytać sygnały radiowe pochodzące z przestrzeni kosmicznej. Po wstępnej obróbce zapisali je do pliku *sygnaly.txt*. W pliku *sygnaly.txt* znajduje się 1000 wierszy. Każdy wiersz zawiera jedno niepuste słowo złożone z wielkich liter alfabetu angielskiego. Długość jednego słowa nie przekracza 100 znaków. Napisz program, który zapisuje podane ciągi tekstowe w tablicy alokowanej dynamicznie typu `char*` o rozmiarze 1000, gdzie pamięć dla poszczególnego ciągu również powinna być alokowana dynamicznie (przydzielona pamięć powinna być dokładnie równa długości danego ciągu liter - pamiętnej o miejscu na znak '\0'). Opisane zadanie tworzenia tablicy i wypełniania jej odpowiednimi danymi z pliku powinno być realizowane przez oddzielną funkcję zwracającą wskaźnik do tej tablicy.

Napisz program, który da odpowiedzi do poniższych podpunktów - do wykonania każdego z podpunktów napisz osobną funkcję. Odpowiedzi zapisz w pliku *wyniki2.txt*, a każdą odpowiedź poprzedź numerem oznaczającym odpowiedni podpunkt.

a) naukowcy zauważyli, że po złączeniu dziesiątych liter co czterdziestego słowa (zaczynając od słowa pierwszego) otrzymamy pewne przesłanie, wypisz to przesłanie (każde co czterdzieste słowo ma co najmniej 10 znaków);

b) znajdź słowo, w którym występuje największa liczba różnych liter; wypisz to słowo i liczbę występujących w nim różnych liter; jeśli słów o największej liczbie różnych liter jest więcej niż jedno, wypisz pierwsze z nich pojawiające się w pliku z danymi;

c) w tym zadaniu rozważmy odległość liter w alfabecie – np. litery `A` i `B` są od siebie oddalone o 1, `A` i `E` o 4, `F` i `D` o 2, a każda litera od siebie samej jest oddalona o 0; wypisz wszystkie

słowa, w których każde dwie litery oddalone są od siebie w alfabecie co najwyżej o 10; słowa wypisz w kolejności występowania w pliku *sygnaly.txt*, po jednym w wierszu; na przykład `CGECF` jest takim słowem, ale `ABEZA` nie jest (odległość `A` – `Z` wynosi 25).