Zadanie: dwa słowa

Marcin osiągnął już wiek, w którym zaczyna uczyć się słów. Dziwnym trafem (no, bo przecież nie w wyniku złośliwości rodziców) wszystkie słowa, jakie do tej pory poznał są tej samej długości. Gdyby znał słowa różnych długości mógłby bawić się w jedną z wielu zabaw, o których słyszał od starszych kolegów. W tej chwili jednak nie pozostaje mu nic innego, jak zająć się jedyną możliwą rozrywką: zamianą liter.

Zabawa polega na tym, że Marcin na początku zapisuje dwa słowa, a potem wybiera po jednej literce z obu słów i... zamienia je miejscami. Po dokonanej zamianie pyta mamę, które słowo jest późniejsze leksykograficznie (sam nawet nie wie, co to słowo oznacza). Mama zawsze bezbłędnie odpowiada na to pytanie, a Marcin kontynuuje swoją zabawę ze zmienionymi już słowami.

Niestety, mama Marcina jest ostatnio bardzo zajęta przygotowywaniem potraw świątecznych. Na szczęście Marcin zawsze może liczyć na swoje starsze rodzeństwo. Zapewne domyślasz się już, że Marcin jest Twoim bratem. Pomóż mu i odpowiedz na jego pytania! Możesz też napisać program, który zrobi to za Ciebie.

Zadanie: dwa słowa

Wejście

Pierwszy wiersz standardowego wejścia zawiera jedną liczbę całkowitą $n \leq 10^6$, oznaczającą długość słów.

Następne dwa wiersze zawierają po jednym słowie długości składające się wyłącznie z małych liter alfabetu angielskiego. Pozycje liter w słowie ponumerowane są od do 0 do $n-1$.

Następny wiersz zawiera jedną liczbę całkowitą $t \leq 10^5$, oznaczającą liczbę pytań Marcina.

W kolejnych wierszach znajdują się po dwie liczby całkowite a i b, mówiące, że przed i-tym pytaniem Marcin zamienił miejscami literę znajdującą się na pozycji a w pierwszym słowie i literę znajdującą się na pozycji b w drugim słowie.

Zadanie: dwa słowa

Wyjście

Standardowe wyjście powinno zawierać dokładnie *t* wierszy. W każdym z nich powinna znajdować się odpowiedź na kolejne pytanie Marcina, zadane po zamianie liter: 0, jeśli po zamianie słowa są równe, 1, jeśli późniejsze leksykograficznie jest słowo pierwsze lub 2, jeśli późniejsze leksykograficznie jest słowo drugie.

Przykład

Dla danych wejściowych:

4
aaab
aaba
2
2 2
3 2

poprawną odpowiedzią jest:

1
0

Wyjaśnienie do przykładu: Po pierwszej zamianie pierwsze słowo ma postać aabb, a drugie aaaa, więc późniejsze leksykograficznie jest słowo pierwsze. Po drugiej zamianie oba słowa mają postać aaba.

Funkcje wejścia / wyjścia

- Znamy już:
  - puts
  - printf
  - scanf
  - getchar

Funkcje wejścia / wyjścia

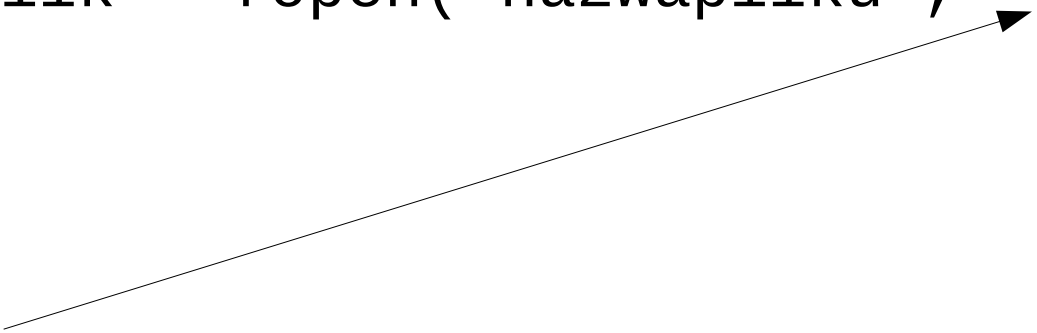
- Znamy już:
  - puts
  - printf
  - scanf
  - getchar
- Cechy wspólne dotychczas poznanych funkcji:
  - puts, printf → standardowe wyjście (stdout)
  - scanf, getchar ← ze standardowego wejścia (stdin)

Funkcje wejścia / wyjścia

- Możemy również czytać / pisać z plików
- FILE* = wskaźnik (uchwyt) na plik
  - FILE = struktura zależna od implementacji

Funkcje wejścia / wyjścia

- Możemy również czytać / pisać z plików
- FILE* = wskaźnik (uchwyt) na plik
  - FILE = struktura zależna od implementacji
- Otwieranie pliku:
  - `FILE* plik = fopen(“nazwapliku”, “rb”);`



Tryb otwarcia pliku:
r = read (do odczytu)
w = write (kasuje zawartość i otwiera do zapisu)
a = append (otwiera do dopisywania na koniec)

Funkcje wejścia / wyjścia

- Możemy również czytać / pisać z plików
- FILE* = wskaźnik (uchwyt) na plik
  - FILE = struktura zależna od implementacji
- Otwieranie pliku:
  - `FILE* plik = fopen(“nazwapliku”, “rb”);`

Tryb otwarcia pliku:
r = read (do odczytu) **r+** także do zapisu
w = write (kasuje zawartość i otwiera do zapisu) **w+** także do odczytu
a = append (otwiera do dopisywania na koniec) **a+** także do odczytu

Funkcje wejścia / wyjścia

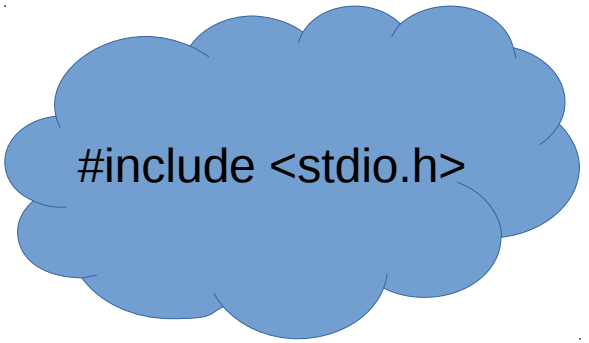
- Możemy również czytać / pisać z plików
- FILE* = wskaźnik (uchwyt) na plik
  - FILE = struktura zależna od implementacji
- Otwieranie pliku:
  - `FILE* plik = fopen(“nazwapliku”, “rb”);`

b oznacza, że chcemy otworzyć plik jako **binarny**,
jeśli tego nie podamy i jesteśmy pod Windows,
otworzymy plik w trybie **tekstowym**:

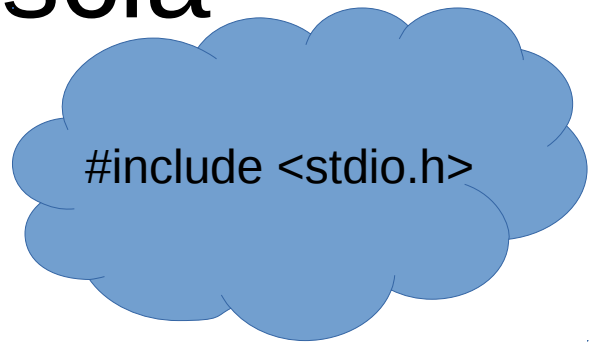
Windowsowe znaki końca linii “\r\n” (13,10) będą odczytywane jako “\n” (10)

Funkcje wejścia / wyjścia

- Możemy również czytać / pisać z plików
- FILE* = wskaźnik (uchwyt) na plik
  - FILE = struktura zależna od implementacji
- Otwieranie pliku:
  - `FILE* plik = fopen(“nazwapliku”, “rb”);`
    `// jeśli się nie uda, zwróci 0 czyli NULL`
- Zamykanie pliku:
  - `fclose(plik);`

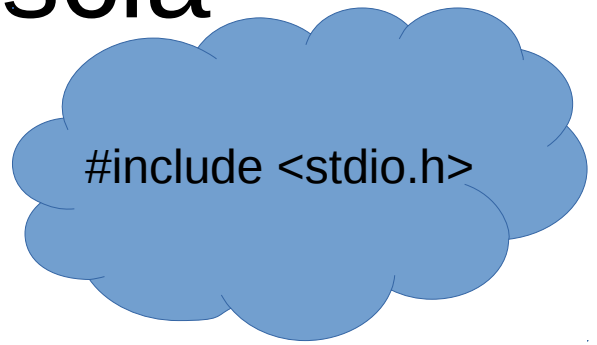


Funkcje wejścia / wyjścia



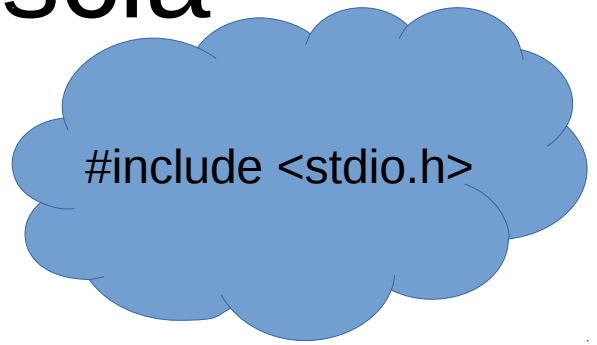
- Czytanie z pliku (binarne)
  - `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);`
- Pisanie do pliku (binarne)
  - `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`

Funkcje wejścia / wyjścia



- Czytanie z pliku (binarne)
  - `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);`
- Pisanie do pliku (binarne)
  - `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`
- Czytanie i pisanie z pliku tekstowe:
  - `int fprintf(FILE *stream, const char *format, ...);`
  - `int fscanf(FILE *stream, const char *format, ...);`
  - `int fgetc(FILE *stream);`
  - `char *fgets(char *s, int size, FILE *stream);`

Funkcje wejścia / wyjścia



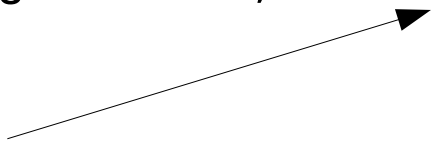
- Czytanie z pliku (binarne)
  - `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);`
- Pisanie do pliku (binarne)
  - `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`
- Czytanie i pisanie z pliku tekstowe:
  - `int fprintf(FILE *stream, const char *format, ...);`
  - `int fscanf(FILE *stream, const char *format, ...);`
  - `int fgetc(FILE *stream); // getchar`
  - `char *fgets(char *s, int size, FILE *stream); // ~~gets~~`

Funkcje wejścia / wyjścia

- Pozycjonowanie wewnątrz pliku:
  - `long ftell(FILE *stream);`
  - `void rewind(FILE *stream);`
  - `int fseek(FILE *stream, long offset, int whence);`

Funkcje wejścia / wyjścia

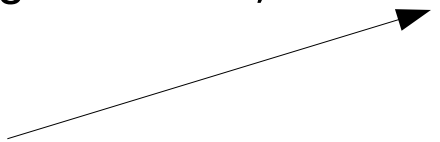
- Pozycjonowanie wewnątrz pliku:
  - `long ftell(FILE *stream);`
  - `void rewind(FILE *stream);`
  - `int fseek(FILE *stream, long offset, int whence);`



SEEK_SET: pozycja względem początku pliku
SEEK_CUR: pozycja względem aktualnej
SEEK_END: pozycja względem końca pliku
(więc offset musi być ujemny)

Funkcje wejścia / wyjścia

- Pozycjonowanie wewnątrz pliku:
  - `long ftell(FILE *stream);`
  - `void rewind(FILE *stream);`
  - `int fseek(FILE *stream, long offset, int whence);`



SEEK_SET: pozycja względem początku pliku
SEEK_CUR: pozycja względem aktualnej
SEEK_END: pozycja względem końca pliku
(więc offset musi być ujemny)

- lub alternatywnie
  - `int fgetpos(FILE *stream, fpos_t *pos);`
  - `int fsetpos(FILE *stream, const fpos_t *pos);`

Funkcje wejścia / wyjścia

- Zapis buforowanych danych:

```
int fflush(FILE *stream);
```

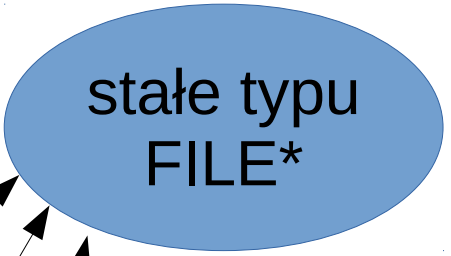
(jest też wykonywany wewnętrznie przez `fclose`)

Funkcje wejścia / wyjścia

- Zapis buforowanych danych:

```
int fflush(FILE *stream);
```

(jest też wykonywany wewnętrznie przez `fclose`)



- Standardowe wejście = **stdin**
- Standardowe wyjście = **stdout**
- Standardowy strumień błędów = **stderr**

(nie jest buforowany)

Zadanie: tekst → HTML

- Program przyjmujący przez wiersz polecenia
  - nazwę pliku wejściowego
  - nazwę pliku wyjściowego

  np. `mojprogram plik.txt nowyplik.html`
- Do nowego pliku program powinien wypisać
  - **<html><body>**
  - treść pliku tekstowego, przy czym pomiędzy kolejnymi liniami ma się znaleźć tag **
**
  - **</body></html>**