Zadanie 15.1

15. Kolokwium C2 - 16/06/2023

1. Kolokwium 16/06/2023 - zadanie na 3,0

Termin 2023-06-16
zostało 3 godz

Trudność
Punkty: 0

Próba
-

Ocena maszyny
Czas testu: 100sek

Inspekcja kodu

Plagiat

Raporty
Raport główny

Przygotuj makro `CREATE_FUNCTION_COUNT01`, które tworzy funkcje zliczającą liczbę wystąpień par bitów `01` w słowie. Funkcja powinna być zbudowana w oparciu o następujący prototyp:

```
uint8_t count01_typ (typ a);
```

gdzie zamiast wyrazu `typ` powinien być wstawiony typ przekazany do makra. Na przykład, dla typu `char` funkcja powinna mieć następujący prototyp:

```
uint8_t count01_char (char a);
```

Deklarację makra zamieść w pliku `utils.h`.

W tym celu zaimplementuj listę **jednokierunkową**, z elementami danymi następującą strukturą:

```
struct node_t {
    char* word_en;
    char* word_pl;
    uint8_t bits;
    struct node_t *next;
};
```

gdzie poszczególne pola mają następujące znaczenie:

- `word_en` - Wskaźnik na słowo, zapisane w danym elemencie listy, przy czym element jest właścicielem bloku danego wskaźnikiem `word_en` (tj. odpowiada za tę pamięć).
- `word_pl` - Wskaźnik na słowo, zapisane w danym elemencie listy, przy czym element jest właścicielem bloku danego wskaźnikiem `word_pl` (tj. odpowiada za tę pamięć).
- `bits` - Suma zliczonych wystąpień par bitów `01` w słowach `word_en` i `word_pl`.
- `next` - Wskaźnik na następny element listy.

Swoją implementację listy oraz funkcje pomocnicze wyposaż w następujące API:

```
int insert(struct node_t **head, const char *word_en, const char *word_pl);
int reverse(struct node_t **head);
void clear(struct node_t **head);
```

Prototypy funkcji należy umieścić w pliku `linked_list.h` a implementacje w pliku `linked_list.c`.

```
int insert(struct node_t **head, const char *word_en, const char *word_pl);
```

Funkcja dodaje słowa `word_en` i `word_pl` do listy jednokierunkowej `head` oraz wyznacza sumę zliczonych wystąpień par bitów `01` w słowach. W celu dodania słów do listy stwórz pomocniczą funkcję o następującej implementacji:

```
struct node_t *create_node(const char *word_en, const char *word_pl, uint8_t (*bits_function)(char));
```

Słowa należy skopiować (kopia głęboka) - dodawany element (a zatem także i lista) jest właścicielem wszystkich swoich danych. Funkcja musi dodać słowa `word_en` i `word_pl` do listy `head` tak, aby ta była zawsze posortowana alfabetycznie rosnąco względem wyrazów w języku angielskim.

Wartość zwracana

Funkcja zwraca następujące wartości:

- `0` - w przypadku pomyślnego dodania nowego elementu do listy,
- `1` - w przypadku błędnych danych, przekazanych do funkcji podczas jej wywoływania,
- `2` - w przypadku stwierdzenia braku pamięci.

Limit pamięci

Funkcja może zaalokować wyłącznie tyle pamięci, ile jest niezbędne do przechowania nowych słów w nowym elemencie oraz samego elementu.

```
void clear(struct node_t **head);
```

Funkcja zwalnia pamięć wszystkich elementów, razem z pamięcią, w której przechowywane są słowa. Następnie funkcja ustawia przekazany jej wskaźnik na `NULL`. W przypadku przekazania niepoprawnych danych funkcja nie wykonuje żadnych działań.

```
int reverse(struct node_t **head);
```

Funkcja odwraca kolejność elementów w liście.

Funkcja nie może alokować żadnej pamięci oraz nie może zmieniać adresów poszczególnych elementów. Odwrócenie elementów musi odbywać się *w miejscu* (warunek sprawdzany przez prowadzącego).

- W przypadku przekazania niepoprawnych danych funkcja nie wykonuje żadnych działań.

Napisz program, który pobierze ciąg słów podanych jako argumenty funkcji `main`, a następnie wstawi je do listy jednokierunkowej, za pomocą funkcji `insert` tak, aby w liście były zawsze posortowane alfabetycznie (od a do z) z pominięciem polskich znaków diakrytycznych (`ęąćńź`...).

Listę należy wyświetlić wprost (od początku do końca - kolejność alfabetyczna), a następnie odwrócić w miejscu (funkcja `reverse`) i ponownie wyświetlić. Przy każdej parze wyrazów należy wyświetlić łączną liczbę ustawionych bitów `01` w bajtach obu wyrazów.

Dane wejściowe należy pobrać z linii poleceń.

- Jeżeli użytkownik nie poda słów podczas uruchamiania program, to należy wyświetlić komunikat błędu `Not enough arguments` i zwrócić kod błędu `9`.

Przykładowe wyjście programu -- sukces:

```
$> ./main gather zbierac matter wzgledu_na_to stand stac receive otrzymac plural liczba_mnoga bar bar head glowa
bar bar 12⏎
gather zbierac 29⏎
head glowa 19⏎
matter wzgledu_na_to 42⏎
plural liczba_mnoga 38⏎
receive otrzymac 35⏎
stand stac 18⏎
stand stac 18⏎
receive otrzymac 35⏎
plural liczba_mnoga 38⏎
matter wzgledu_na_to 42⏎
head glowa 19⏎
gather zbierac 29⏎
bar bar 12⏎
```

Przykładowe wyjście programu -- brak danych:

```
$> ./main
Not enough arguments⏎
```

Uwagi

- W programie nie wolno używać unii oraz operatorów `%` i `[]`.
- Podczas uruchamiania funkcji `main` system Dante wykorzystuje pierwszy parametr uruchamianego programu. W przypadku testów jest on równy `0` i nie jest dostarczany do testowanej funkcji `main`. Zapis `./main 0 4` oznacza uruchomienie programu z jednym

parametrem: 4.