

1. Znaki i stringi

1 Zadanie

1.1 Zliczanie linii, słów i znaków

Szablon programu należy uzupełnić o definicję funkcji `wc()`, która czyta tekst ze standardowego wejścia i na tej podstawie zlicza linie, słowa oraz znaki, występujące w tym tekście, podobnie jak komenda `wc` systemu Unix. Słowo to ciąg znaków oddzielony spacją, tabulatorem lub znakiem nowej linii.

Opis komendy `wc`: [https://en.wikipedia.org/wiki/Wc_\(Unix\)](https://en.wikipedia.org/wiki/Wc_(Unix))

- **Wejście**

1
linie tekstu

- **Wyjście**

Liczba linii, słów i znaków w tekście

- **Przykład:**

Wejście:

```
1
int main() {
    printf ("Hello\n");
    return 0;
}
```

Wyjście: 4 8 47

1.2 Liczności znaków

Szablon programu należy uzupełnić o definicję funkcji `char_count()`, która czyta tekst ze standardowego wejścia i na tej podstawie zlicza krotności znaków występujących w tym tekście.

Rozpatrujemy znaki należące do przedziału `[FIRST_CHAR, LAST_CHAR-1]`. Powyższe stałe są zdefiniowane w szablonie programu.

Funkcja następnie sortuje liczności znaków malejąco (sortujemy indeksy a nie samą tablicę zliczającą) i zwraca, poprzez parametry `n_char` i `cnt`, `char_no`-ty (co do liczności) znak tekstu oraz liczbę jego wystąpień. W przypadku jednakowej liczności znaki powinny być posortowane alfabetycznie.

- **Wejście**

2

`char_no`

linie tekstu

- **Wyjście**

`char_no`-ty najliczniejszy znak i liczba jego wystąpień

- **Przykład:**

Wejście:

2

1

```
int main() {  
    printf ("Hello\n");  
    return 0;  
}
```

Wyjście: n 5

1.3 Zliczanie digramów

Szablon programu należy uzupełnić o definicję funkcji `digram_count()`, która czyta tekst ze standardowego wejścia i na tej podstawie zlicza krotności digramów znakowych (par znaków) występujących w tym tekście.

Rozpatrujemy znaki należące do przedziału `[FIRST_CHAR, LAST_CHAR-1]`. Powyższe stałe są zdefiniowane w szablonie programu.

Funkcja następnie sortuje licznosci digramów malejąco (sortujemy indeksy a nie samą tablicę zliczającą) i zwraca, poprzez parametry `n_char` i `cnt`, `digram_no`-ty (co do licznosci) digram tekstu oraz liczbę jego wystąpień. W przypadku jednakowej licznosci digramy powinny być posortowane alfabetycznie.

- **Wejście**

```
3
digram_no
linie tekstu
```

- **Wyjście**

`digram_no`-ty najliczniejszy digram (dwa znaki bez spacji) i liczba jego wystąpień

- **Przykład:**

Wejście:

```
3
1
int main() {
    printf ("Hello\n");
    return 0;
}
```

Wyjście: in 3

1.4 Zliczanie komentarzy

Szablon programu należy uzupełnić o definicję funkcji `find_comments()`, która czyta ze standardowego wejścia ciąg znaków stanowiący program w języku C. Funkcja zlicza komentarze blokowe (`/* ... */`) i jednoliniowe (`// ...`) w przeczytanym tekście i zwraca uzyskane liczby do funkcji `main()` przy użyciu parametrów.

Zagnieżdżone komentarze nie są liczone, czyli np. następujący fragment kodu:

```
/*// tekst*/
```

jest uważany za jeden komentarz blokowy.

Można założyć, że wszystkie komentarze blokowe są prawidłowo zamknięte.

- **Wejście**

4
linie tekstu

- **Wyjście**

Liczba komentarzy blokowych i liniowych

- **Przykład:**

Wejście:

```
4
int main() { // comment
    printf ("Hello\n"); /* and another */
    return 0;
    /* and more
    and more ...
    */
}
```

Wyjście: 2 1