4 Operacje na znakach odczytywanych ze stdin

Wskazówka dotycząca strumienia danych wejściowych:
Stała `TEST` jest przełącznikiem między pracą w trybie testowania programu (= 1), a wersją przygotowaną do automatycznej oceny (=0, pomija wyprowadzanie komunikatów dla użytkownika).

W zadaniach tego rozdziału stała ta pozwala przełączać strumień wejściowy programu między wejście standardowe (klawiaturę) `stdin` i plikiem wskazanym przez użytkownika.

- Gdy `TEST` = 1, linie tekstu są wczytywane z klawiatury aż do znaku Ctrl+D (UNIX) albo Ctrl+Z (Windows).
- Gdy `TEST` = 0, ale program nie jest przesyłany do oceny automatycznej, to po wyborze zadania (np. 3) należy wpisać nazwę pliku zawierającego tekst (oraz ewentualne dalsze parametry zadania).

Plik zawierający dane testowe należy utworzyć obok pliku z źródłową wersją pisanego programu (w opcji Edycja, po rozwinięciu menu ikonką + pojawia się ikonka dodawania nowego pliku).

4.1 Zadanie 4.1

4.1.1 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)

- **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

4.1.2 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

4.1.3 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 liczności digramów malejąco (sortujemy indeksy a nie samą tablicę zliczającą) i zwraca, poprzez parametry `n_char` i `cnt`, `digram_no`-ty (co do liczności) digram tekstu oraz liczbę jego wystąpień. W przypadku jednakowej liczności 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

4.1.4 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

4.2 Zadanie 4.2

4.2.1 Znajdowanie identyfikatorów

Szablon programu należy uzupełnić o definicję funkcji `find_idents()`, która czyta ze standardowego wejścia ciąg znaków stanowiący program w języku C. Funkcja zlicza **unikalne** identyfikatory zawarte w przeczytanym tekście i zwraca uzyskaną liczbę do funkcji `main()`.

Definicja identyfikatora jest taka jak w języku C, czyli jest to ciąg liter i cyfr zaczynający się od litery; znak podkreślnika jest uważany za literę, czyli na przykład identyfikator `_a` jest legalny.

Uwaga 1: W programie wejściowym mogą znajdować się stałe znakowe, napisowe (ciągi znaków ujęte w cudzysłowy) oraz komentarze (jak w zadaniu o komentarzach). Powinny one być pomijane przy liczeniu identyfikatorów.

Uwaga 2: Dla ujednolicenia templatka programu zawiera słowa kluczowe, które należy wyłączyć z listy identyfikatorów.

- **Wejście**
  `linie tekstu`
- **Wyjście**
  liczba unikalnych identyfikatorów zawartych w tekście nie będących słowami kluczowymi
- **Przykład:**
  Wejście:

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

  Wyjście: 2