Zadanie G - Pliki

Punktów procentowych do uzyskania: **10**

Język programowania: C++

autor zadania: Rafał Kawa

Opis zadania

Zadaniem jest stworzenie programu działającego na plikach tekstowych zawierających w pierwszej linii liczbę pamiętanych rekordów danych zaś w kolejnych liniach o liczności równej początkowej liczbie pamiętających pojedyncze rekordy obejmujące oddzielone pojedynczą spacją następujące informacje:

1. *`ident`* (identyfikator) - w postaci ciągu znakowego nie zawierającego spacji.
2. *`height`* (wysokość) - liczba typu **`float`**.
3. *`age`* (wiek) - liczba typu **`int`**.

Wejście

- Pierwsza linia wejścia zawiera całkowitą liczbę dodatnią będącą liczbą koniecznych do wykonania poleceń.
- Każde polecenie umieszczone jest w jednej linii i składa się z nazwy polecenia oraz od dwóch do pięciu parametrów oddzielonych pojedynczą spacją.
- Wybrane polecenia w zależności od parametrów mogą wymagać występujących tuż po linii polecenia dodatkowych linii wejścia.

Opis poleceń

Opis poleceń będzie posługiwał się pojęciami:

– pozycja rekordu w pliku - czyli numer rekordu od początku pliku począwszy od wartości 0.
– wyświetlenia rekordu - oznacza wyświetlenie w jednej linii wartości jego pozycji w pliku z następującymi znakiem dwukropka i znakiem spacji, po których następują składowe rekordu oddzielone pojedynczą spacją w kolejności zgodnej ze specyfikacją rekordu w opisie zadania.

- `create` *`fileName amount`*
  Tworzy plik o nazwie *`fileName`* zawierający liczbę rekordów danych wynoszącą *`amount`*, których zawartość jest podana w następujących liniach wejścia w ilości równej *`amount`* zgodnie z kolejnością opisu rekordu i oddzielonych pojedynczymi spacjami.
- `display` *`fileName position amount`*
  Dla pliku o nazwie *`fileName`* wyświetla rekordy w liczbie wynoszącej *`amount`* począwszy od pozycji równej *`position`*. Ponadto, po wyświetleniu wszystkich wymaganych rekordów wypisywana jest linia z pojedynczą spacją.
- `insert` *`fileName position amount`*
  Do pliku o nazwie *`fileName`* począwszy od pozycji *`position`* wstawia rekordy w liczbie *`amount`*, z opisami umieszczonymi w kolejnych wierszach w liczbie równej *`amount`* analogicznie jak dla polecenia `create`. Możliwe jest użycie nie więcej niż jednego pliku pomocniczego o nazwie koniecznie `tmp.txt`, który o ile będzie tworzony musi być na końcu implementacji polecenia usunięty konieczną instrukcją
  `remove ( "tmp.txt" );`
- `delete` *`fileName position amount`*
  Z pliku o nazwie *`fileName`* usuwa rekordy w liczbie danej parametrem *`amount`* począwszy od pozycji danej parametrem *`position`*. Możliwe jest użycie nie więcej niż jednego pliku pomocniczego o nazwie koniecznie `tmp.txt`, który o ile będzie tworzony musi być na końcu implementacji polecenia usunięty konieczną instrukcją
  `remove ( "tmp.txt" );`
- `search` *`fileName searchIdent searchAge startHeight finishHeight`*
  W pliku o nazwie *`fileName`* znajduje a następnie wyświetla rekordy, których identyfikator jest równy *`searchIdent`*, wiek jest równy *`searchAge`* natomiast wysokość mieści się w zakresie wartości od *`startHeight`* do *`finishHeight`*. Podanie znaku * w miejscu *`searchIdent`* powoduje zaniechanie wymogu zgodności identyfikatora, zaś podanie wartości `0` w miejscu *`searchAge`* powoduje zaniechanie wymogu zgodności wieku. Samo wyświetlanie jest zgodne z opisanym dla polecenia `display`.
- `sort` *`inputFileName outputFileName`*
  Wszystkie rekordy z pliku o nazwie *`inputFileName`* są przenoszone do pliku o nazwie *`outputFilename`* tak, by były posortowane niemalejąco z punktu widzenia identyfikatora. Kolejność rekordów o równych identyfikatorach w pliku wejściowym musi być utrzymana w pliku wyjściowym, zaś kasowanie rekordów z pliku wejściowego może się odbywać tylko poprzez zaimplementowane wcześniej polecenie `delete`.
- `merge` *`prevFileName nextFileName mergeFileName`*
  Rekordy z plików *`prevFileName`* oraz *`nextFileName`* są kopiowane (bez kasowania) do pliku o nazwie *`mergeFileName`*. Zakładamy, że rekordy w plikach *`prevFileName`* oraz *`nextFileName`* są posortowane niemalejąco względem identyfikatora a zarazem wymagane jest analogicznie sortowanie rekordów w pliku o nazwie *`mergeFileName`*. Kolejność rekordów o równym identyfikatorze w plikach *`prevFileName`* oraz *`nextFileName`* musi pozostać niezmienna, natomiast w przypadku jednakowych identyfikatorów w plikach o nazwach *`prevFileName`* oraz *`nextFileName`* pierwsze zapisywane są rekordy z pliku o nazwie *`prevFileName`*.

Dodatkowe uwarunkowania

- Jedynymi dopuszczalnymi do włączenia plikami nagłówkowym są `cstdio` (wyłącznie dla instrukcji `remove`), `fstream`, `iostream` oraz `string`.
- Dla nazw plików dopuszczalne (a nawet zalecane) jest posługiwanie się metodą `c_str ()` zmiennej typu `string` lub literałami ciągowoznakowymi.
- Zabronione jest używanie znaków kwadratowych nawiasów i ogólnie tablic.
- Zabroniony jest dynamiczny przydział pamięci.
- **Użycie metod `seekp` oraz `seekg` jest zabronione.**
- **Plik z rozwiązaniem musi nazywać się `source.cpp` i wysłany na bacę musi być spakowany programem zip.**

Przykłady wejścia i odpowiadającego wyjścia

wejście	wyjście
6 create file.txt 3 alpha 1.1 44 beta 2 55 gamma 3.33 66 display file.txt 0 3 insert file.txt 1 2 delta 77.7 99 epsilon 88.8 100 display file.txt 0 4 delete file.txt 2 2 display file.txt 0 8	0: alpha 1.1 44 1: beta 2 55 2: gamma 3.33 66 0: alpha 1.1 44 1: delta 77.7 99 2: epsilon 88.8 100 3: beta 2 55 0: alpha 1.1 44 1: delta 77.7 99 2: gamma 3.33 66

wejście	wyjście
6 create prev.txt 4 alpha 1.1 1 delta 3.3 3 delta 2.2 4 gamma 7.7 7 create next.txt 3 beta 2.2 2 delta 1.1 5 epsilon 3.3 6 merge prev.txt next.txt merge.txt display prev.txt 0 3 display next.txt 0 3 display merge.txt 0 6	0: alpha 1.1 1 1: delta 3.3 3 2: delta 2.2 4 0: beta 2.2 2 1: delta 1.1 5 2: epsilon 3.3 6 0: alpha 1.1 1 1: beta 2.2 2 2: delta 3.3 3 3: delta 2.2 4 4: delta 1.1 5 5: epsilon 3.3 6

wejście	wyjście
8 create inFile.txt 8 beta 3.3 3 beta 3.3 2 beta 2.2 3 alpha 2.2 2 alpha 1.1 2 alpha 2.2 1 beta 2.2 2 alpha 1.1 1 search inFile.txt alpha 1 1.0 2.3 search inFile.txt alpha 0 1.1 1.1 search inFile.txt * 2 1.1 2.2 search inFile.txt * 0 1.2 3.4 sort inFile.txt outFile.txt display inFile.txt 0 1 display outFile.txt 0 8	5: alpha 2.2 1 7: alpha 1.1 1 4: alpha 1.1 2 7: alpha 1.1 1 3: alpha 2.2 2 4: alpha 1.1 2 6: beta 2.2 2 0: beta 3.3 3 1: beta 3.3 2 2: beta 2.2 3 3: alpha 2.2 2 5: alpha 2.2 1 6: beta 2.2 2 0: alpha 2.2 2 1: alpha 1.1 2 2: alpha 2.2 1 3: alpha 1.1 1 4: beta 3.3 3 5: beta 3.3 2 6: beta 2.2 3 7: beta 2.2 2