Politechnika Śląska w Gliwicach
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Lister

autor	Szymon Stasiak
prowadzący	dr inż. Wojciech Sułek
rok akademicki	2019/ 2020
kierunek	Teleinformatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium / ćwiczeń	czwartek, 14:15 – 15:45
grupa	3
sekcja	9
termin oddania sprawozdania	2020-01-30
data oddania sprawozdania	2020-01-30

1 Treść zadania

Napisać program, który wczytuje plik tekstowy i wypisuje w pliku wynikowym wszystkie słowa pliku w kolejności alfabetycznej wraz z numerami linii, w których słowo to występuje. -i plik wejściowy z liczbami do posortowania -o plik wyjściowy z liczbami posortowanymi.

`-i` plik wejściowy z liczbami do posortowania
`-o` plik wyjściowy z liczbami posortowanymi

2 Analiza zadania

Zagadnienie przedstawia probelm czytania i zapisywania plików, oraz ich analizę. Poprzez wczytanie pliku, dążymy do zapisania linii, w których owe słowa występują.

2.1 Struktury danych

W programie wykorzystano drzewo binarne do przechowywania wartości. Drzewo binarne przechowuje dane w węzłach. Węzeł może mieć od 0 do 2 potomków, przy czym po lewej stronie węzła znajdują się potomki przechowujące wartości nie większe niż węzeł rodzicielski, po prawej zaś większe.

2.2 Algorytmy

W programie wykorzsytano listy do przechowywania słów oraz numerów linii. Rozmiar list jest ograniczony pojemnością pamieci. Każdy węzeł listy zawiera słowo oraz listę linii, gdzie każdy węzeł tej listy zawiera numer linii.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: -i dla pliku wejściowego i -o dla pliku wyjściowego), np. program -i wejscie.txt -o wyjscie.txt program -o wyjscie.txt -i wejscie.txt

```
program -i wejście.txt -o wyjście.txt
program -o wyjście.txt -i wyjście.txt
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. Wszystkie użyte struktury zostały wydzielone na pliki nagłówkowe i źródłowe.

4.1 Typy zdefinowane w programie

W programie zdefiniowano następujący typ:

```
1  struct word
2  {
3      std::string phrase;
4      line* lineList;
5      word* nextWord;
6
7      void addWord(std::string phrase, int lineNumber);
8      void displayCurrent();
9      void displayAll();
10     void clear();
11     word();
12 };
```

Typ ten służy do zbudowania listy słów.

Zdefiniowano także typ do zapisywania informacji o liniach..

```
1  struct line
2  {
3      int lineNumber;
4      line* next;
5      void clear();
6      line(int lineNumber);
7  };
```

4.2 Ogólna struktura programu

W funkcji głównej sprawdzane są parametry wejściowe

```
1  if (argc != 5)
2      {
3          std::cout << "Nalezy␣podac␣cztery␣argumenty.␣-i␣<
               plik_wejsciowy>␣-o␣<plik_wyjsciowy>";
```

```
4         return -1;
5     }
```

Następnie jest sprawdzane, w którym parametrze wejściowym jest zapisana nazwa pliku wejściowego oraz wyjściowego.

```
1 if (firstArgument == "-i")
2         inputFile = 2;
3     if (secondArgument == "-i")
4         inputFile = 4;
5     if (firstArgument == "-o")
6         outputFile = 2;
7     if (secondArgument == "-o")
8         outputFile = 4;
```

Po sprawdzeniu parametrów w zmiennych nastepuje wczytanie plików i zapisanie słów w liście. Następnie wywoływana jest funkcja:

```
1  readFile(argv[ inputFile ]) ;
```

Funkcja ta otwiera plik wejściowy, sczytuje słowa i umieszcza je w liście o korzeniu base. Po sczytaniu wszystkich list funkcja zamyka plik. Następnie wywoływana jest funkcja:

```
1 base->displayAll();
```

Funkcja ta wyświetla wszystkie słowa z listy, oraz numery lini i ich wystapień. Ostatnią funkcją programu jest funkcja zwalniająca pamięć:

```
1 base->clear();
2 delate (base);
```

4.3 Szczegółowy opis implementacji

```
1 void readfile (std::string fileName)
```

Funkcja otwiera plik, czyta kolejne linie i dodaje słowa do listy. Funkcja otwiera plik po nazie przekazanej w parametrze fileName. Wszystkie przeczytane słowa zsotają dodane do base.

Funkcja displayAll wyświetlawszystkieelementylisty.Pokażdymsłowie wypisne są wszystkie numery linii, w których występują. Wyświetlanie jest zaimplementowane z użyciem rekurencji.

```
1 void word::displayAll()
2 {
3     this->displayCurrent();
4     if (this->nextWord != 0)
5     {
6         this->nextWord->displayAll();
7     }
8 }
```

Funkcja saveResultToFile zapisuje wszystkie elementy listy. Po każdym słowie wypisne są wszystkie numery linii, w których występują. Każde słowo jest w nowej linii.

```
1 void saveResultToFile(std::string fileName)
2 {
3     std::fstream   outFile;
4     outFile.open(fileName, std::fstream::out);
5
6     word* temp = base; //wskazanie na poczatek listy
          slow
7     while (temp != 0)
8     {
9         std::string word = temp->phrase;
10        line* lineList = temp->lineList;
11        outFile << word << " : ";
12        while (lineList != 0)
13        {
14            outFile << lineList->lineNumber << ", ";
15            lineList = lineList->next;
16        }
17        outFile << std::endl;
18        temp = temp->nextWord;
19    }
20    outFile.close();
21 }
```

Ostatnia funkcja wykonywana w programie odpowiada za czyszczenie pamięci. Do usunięcia wszystkich węzłów używana jest rekurencja.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach wejściowych. Były używane pliki zarówno w języku polskim jak i angielskim, jednak jeżeli słowo w języku polskim zaczyna się od wielkiej litery polskiej, program nie jest w stanie go posortować.

6 Wnioski

Program do listowania słów jest programem złożonym, ponieważ wymaganie samodzielnego zarządzania pamięcią. Najbardziej wymagające okazało dodawanie słów, aby były zapisane w kolejności alfabetycznej. Szczególnie trudne było zapewnienie działania programu bezpośrednio z konsoli. Dla pewnych danych program wykonywał się poprawnie na wszystkich komputerach, które były użyte do testów.