Politechnika Śląska w Gliwicach

Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Temat: Tłumacz

Autor	Damian Tabaka
Prowadzący	Mgr inż. Anna Kuliś
Rok Akademicki	2017/2018
Kierunek	Teleinformatyka
Rodzaj studiów	SSI
Termin laboratorium/ćwiczeń	czwartek, 10:00 – 11:30
Grupa	2
Sekcja	8
Termin oddania sprawozdania	2018-01-25
Data oddania sprawozdania	2018-01-25

1 Treść zadania

Napisać program, który tłumaczy tekst z języka polskiego na angielski i z powrotem. Tłumaczenie polega na zastępowaniu słów języka wyjściowego słowami języka docelowego. Jest to zatem sposób dość niedoskonały. Przy implementacji należy skorzystać z drzewa binarnego. Drzewo binarne zbudowane dla języka polskiego zbudowane jest w ten sposób, że wyrazy, które stoją wcześniej w porządku alfabetycznym są po lewej stronie węzła, a te, które stoją później – po prawej stronie. Te same elementy, które należą do drzewa dla języka polskiego, są także elementami drzewa dla języka angielskiego budowanego według porządku alfabetycznego wyrazów angielskich. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

-s plik wejściowy słownika

-i plik wejściowy z tekstem do przetłumaczenia

-o plik wyjściowy z przetłumaczonym tekstem

-k kierunek tłumaczenia

Po przełączniku -k może wystąpić jedna z dwóch wartości: enpl (tłumaczenie z j. angielskiego na polski) albo plen (tłumaczenie z j. polskiego na angielski).

2 Analiza zadania

Zagadnienie przedstawia problem, przetłumaczenia tekstu podanego w plikach z języka polskiego na angielski i z powrotem.

2.1 Struktury danych

W programie wykorzystano strukturę drzewa binarnego do przechowywania wyrazów. Drzewo przechowuje dane w węzłach. Przy czym po lewej stronie znajdują się potomki przechowujące wyrazy mniejsze w porządku alfabetycznym, nie większym niż korzeń z kolei po prawej większe. Drzewo osobno wskazuje kolejne drzewa dla wyrazów angielskich i polskich. Struktura dzięki swoim możliwością szybkiego sortowania, idealnie pasuje do tego typu problemu.

2.2 Algorytmy

Program sortuje wyrazy poprzez umieszczanie ich w drzewie binarnym. Dodawane wyrazy są sortowane rekurencyjnie pod względem porządku alfabetycznego odpowiednio wstawiając je do stron drzewa binarnego.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego, wyjściowego oraz słownikowego a także kierunek tłumaczenia (plen, enpl) zaraz po odpowiednich parametrach dla każdego z nich, (parametr -i dla pliku wejściowego, -o dla pliku wyjściowego oraz –s dla pliku słownikowego a także –k parametr dla kierunku tłumaczenia). Kolejność przełączników jest obojętna, a błędne wpisanie, brak parametrów bądź nieodpowiednie nazwy plików powodują wyświetlenie się komunikatu o błędzie oraz wyświetla pomoc z przykładowym prawidłowym wywołaniem funkcji. Przykładowe wywołania:

```
Project1.exe -o plikwy.txt -k enpl -s slownik.txt -i plikwe.txt
Project1.exe -o plikwy.txt -k plen -i plikwe.txt -s slownik.txt
```

Przykład komunikatu błędu dla braku bądź nieodpowiedniego parametru:
`Project1.exe - plikwy.txt -k enpl -s slownik.txt -i plikwe.txt`
`„BRAK PARAMETRU PLIKU WYJSCIA!”`

Komunikat pomocy wyświetlany, gdy zostaną wpisane złe parametry:
`„Jak uruchomic program: -o plikwy.txt -k enpl -s slownik.txt -i plikwe.txt”`

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikacje z użytkownikiem) od logiki aplikacji (sortowania liter).

4.1 Typy zdefiniowane w programie

W programie zdefiniowano następujący typ drzewa binarnego:

```
struct drzewo{
      string pol;       //Polskie wartości przechowywane w węźle
      drzewo *pright;   //potomek wezla
      drzewo *pleft;    //potomek wezla
      string ang;       //Angielskie wartości przechowywane w węźle
      drzewo *aright;   //potomek wezla
      drzewo *aleft;    //potomek wezla
};
```

4.2 Ogólna struktura programu

W funkcji głównej wywoływana jest funkcja:

translate(spri, spro, sprs, sprk);

Która uruchamia się w momencie gdy program główny został wywołany w prawidłowy sposób, to znaczy, że wszystkie pliki oraz parametry są wpisane prawidłowo, po sprawdzeniu wszystkich parametrów w zmiennych spri, spro, sprs, sprk przechowywane są odpowiednio nazwy pliku wejściowego, wyjściowego, słownikowego oraz kierunku tłumaczenia, następnie powyższa funkcja tłumaczy tekst z pliku wejściowego do wyjściowego, po czym program się kończy. Jeśli jednak parametry są wpisane w sposób nieprawidłowy, to zostaje wyświetlany komunikat z błędem i program kończy swoją prace przed uruchomieniem funkcji translate.

4.3 Szczegółowy opis implementacji funkcji

void translate(spri, spro, sprs, sprk);

Funkcja, która tłumaczy dane z pliku wejściowego spri i zapisuje je do pliku wejściowego spro. Kierunek tłumaczenia określany jest przez zmienna sprk. Dane z pliku słownikowego zapisywane są drzewa binarnego.

void pomoc();

```
void pomoc() {
cout << "Jak uruchomic program:" << endl << "-o plikwy.txt -k enpl -
s slownik.txt -i plikwe.txt";
}
```

Funkcja pomoc wyświetlana jest gdy wpisywane są złe dane podczas uruchamiania programu. Informuje nas jak w poprawny sposób uruchomić program.

void DODWA(drzewo *& glowa, drzewo *& y);

```
void DODWA(drzewo *& glowa, drzewo *& y)
                {
     if (glowa != nullptr && y != nullptr) {
                if ((y->ang) > (glowa->ang)) {
                     if (glowa->aleft == nullptr) {
                          glowa->aleft = y;
     }
     else
     {
```

```
            DODWA(glowa->aleft, y);
            }
      }
      else
            if (glowa->aright == nullptr) {
                        glowa->aright = y;
      }
      else
      {
            DODWA(glowa->aright, y);
                  }
      }
}
```

Funkcja, która dodaje do drzewa binarnego o głowie glowa wyrazy angielskie, które są wskazywane przez y. Najpierw sprawdzamy czy miejsce gdzie jest angielski wyraz i głowa nie są nullptr, jeśli nie to sprawdzamy kolejność alfabetyczną, następnie czy to jest jeszcze kolejny element drzewa, na koniec umieszczamy wyraz w odpowiednim węźle.

void DeleteEng(drzewo *& glowa);

```
void DeleteEng(drzewo *& glowa) {
                  if (glowa != nullptr) {
            DeleteEng(glowa->aleft);
            DeleteEng(glowa->aright);
      delete glowa;
      }
}
```

Funkcja, która usuwa drzewo binarne o głowie glowa w celu zwolnienia pamięci programu.

string plen(drzewo *& glowa, const string & pl);

```
string plen(drzewo *& glowa, const  string & pl) {
      if (glowa != nullptr) {
            if (pl == glowa->pol) {
```

```
            return glowa->ang;
        }
        if (pl > glowa->pol) {
            return plen(glowa->pleft, pl);
    }
    else
    {
            return plen(glowa->pright, pl);
    }
    }
    else
    {
        string en;
        en = "[" + pl + "]";
        return en;
    }
}
```

Funkcja która wyszukuje tłumaczenia w drzewie binarnym o głowie głowa słów polskich, później następuje ich zamiana, wszystko w porządku alfabetycznym, jeśli nie posiadamy tłumaczenia wykonuje, podany wyraz obejmujemy w nawiasy kwadratowe.

string enpl(drzewo *& glowa, const string & en);

```
string enpl(drzewo *& glowa, const string & en) {
    if (glowa != nullptr) {
        if (en == glowa->ang) {
            return glowa->pol;
    }
        if (en > glowa->ang) {
            return enpl(glowa->aleft, en);
    }
    else
    {
            return enpl(glowa->aright, en);
```

```
	}
	}
	else
	{
		string pl;
		pl = "[" + en + "]";
		return pl;
	}
}
```

Funkcja która wyszukuje tłumaczenia w drzewie binarnym o głowie głowa słów angielskich, później następuje ich zamiana, wszystko w porządku alfabetycznym, , jeśli nie posiadamy tłumaczenia wykonuje, podany wyraz obejmujemy w nawiasy kwadratowe.

drzewo* DODW(drzewo *&glowa, const string & pol, const string & ang);

```
drzewo*  DODW(drzewo *&glowa, const string & pol, const string &
ang){
		if (glowa == nullptr) {
			glowa = new drzewo;
		glowa->pol = pol;
		glowa->ang = ang;
		glowa->pright = nullptr;
		glowa->pleft = nullptr;
		glowa->aright = nullptr;
		glowa->aleft = nullptr;
		return glowa;
	}
	else
	{
		if (pol > (glowa->pol)) {
		return DODW(glowa->pleft, pol, ang);
	}
	else
```

```
        {
                return DODW(glowa->pright, pol, ang);
            }
        }
}
```

Funkcja dodaje na drzewo binarne wyrazy polskie i angielskie. Zwraca wskazanie na nowo utworzoną część drzewa.

6 Testowanie

Program został przetestowany na różnego rodzajach plikach. Jak na przykład plik wyjścia (do którego zapisujemy wynik programu) z rozszerzeniem „txt” bądź dokument word z rozszerzeniem „docx” oba zapisy działają prawidłowo. Program został również poddany próbie, podczas wpisywania parametrów. Wszystko działa poprawnie, jeśli zabraknie parametru, zostanie źle wpisany bądź brakuje jakiegoś pliku zostanie wyświetlony stosowny komunikat o problemie. Program został sprawdzony pod kątem wycieków pamięci.

7 Wnioski

Program tłumacz wymagał dużo pracy oraz poświęcenia czasu, jednak udało się go ukończyć, największym problemem było zrozumienie struktury drzewa. Na początku program sprawiał duże problemy jednak nauczyłem się nowych rzeczy jak np. sprawdzanie wycieków pamięci.