**Садов Д.В. ИВБО 7-14.**

**Лабораторная работа №5** В данной лабораторной работе нужно было выполнить следующее:

- ограничить по времени считывание слов из файла

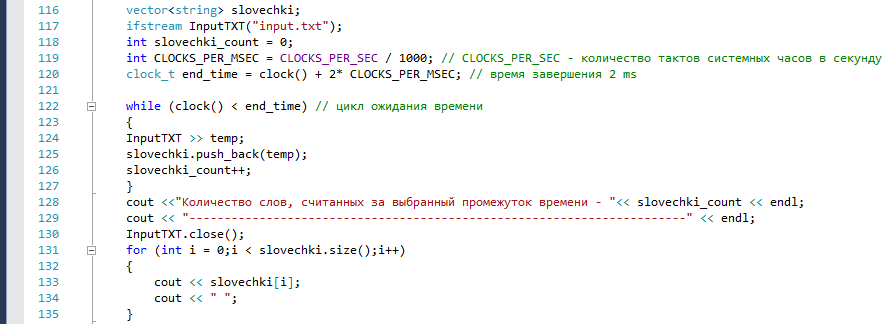
- определить в каких считанных словах были произведены ошибки (например присутствует число посреди слова)

- промаркировать корректные слова и слова с ошибками и разделить на 2 массива

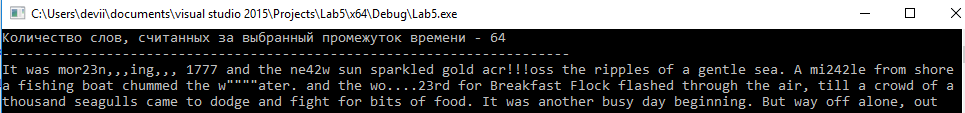
- слова с ошибками исправить, а затем склеить с корректными, так чтобы порядок в тексте был не нарушен.

1. Исходный текст находится в файле input.txt

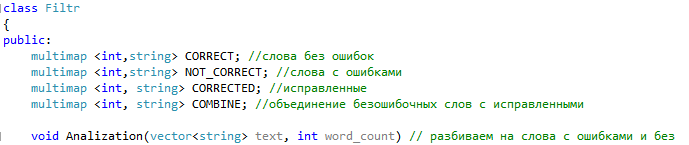
Чтобы ограничить по времени число считываемых слов, в программе было реализовано следующее:



Промежуток выбран 2мс, в результате чего пользователь видит у себя на экране следующее:

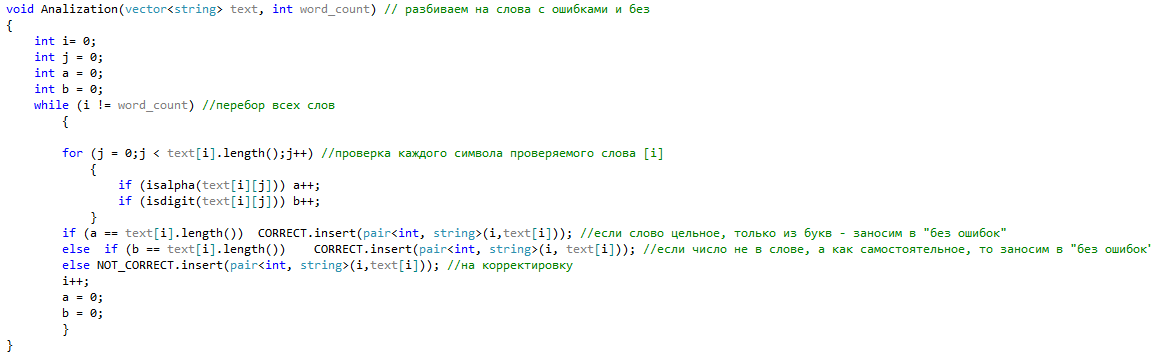


1. Далее был реализован класс FIltr со следующими полями и методами:





1. После того, как мы считали за промежуток времени некоторое количество слов, в метод Analization передается считанные слова и их общее количество. Сам метод выглядит следующим образом:

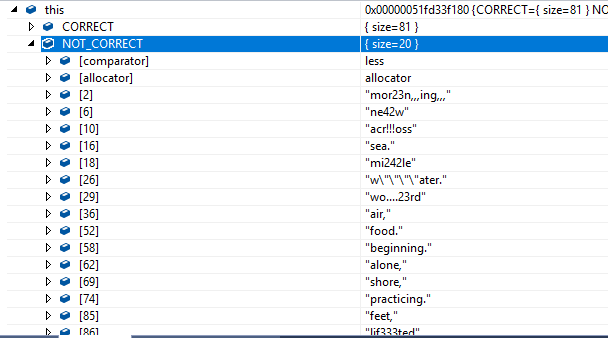


Если слово состоит только из букв, мы засовываем его в контейнер CORRECT, в поле ключа заносим айди под которым просматривалось слово, а в поле значения- само слово.

То же самое делаем если число цельное, то есть не находится посреди букв.

Остальные слова записываются в контейнер NOT\_CORRECT

Так например выглядит контейнер со словами, которые нужно исправить :



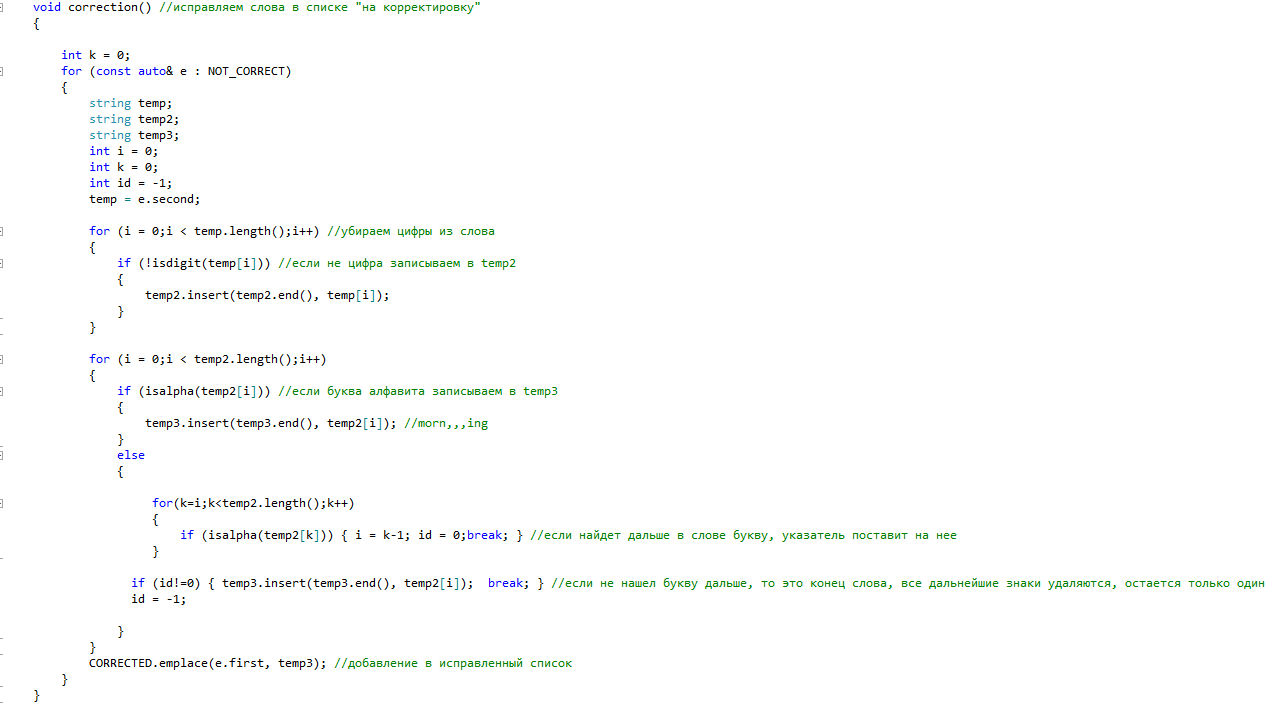
1. После того, как слова разделены, используя метод correction исправляем слова в контейнере NOT\_CORRECT.

Алгоритм метода заключается в следующем:

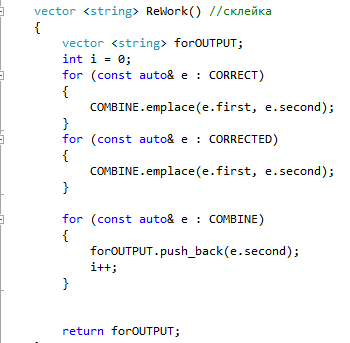
1. Убираются из слова цифры
2. Далее решается вопрос со знаками. Считывались слова вместе со знаками до пробела, т.е. в предложении Я, иду гулять – 3 слова [Я,] [иду] [гулять]

Если программа видит знак, а не букву, то она смотрит есть ли дальше хотя бы одна буква в слове. Если находит букву, то знак/знаки до найденной буквы удаляются и указатель становится на найденную букву. Если буква не найдена, а присутствуют дубликаты знаков, например много запятых ,,,,,, то оставляется один символ, остальные удаляются. После корректировки, слова записываются в контейнер CORRECTED.

Сам метод выглядит так:

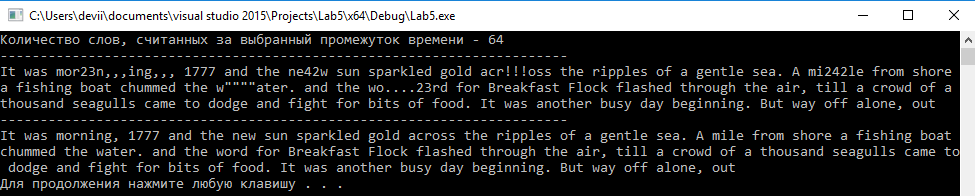


1. Операция склеивания осуществляется методом ReWork



Данный метод объединяет слова из 2ух контейнеров CORRECT и CORRECTED в один общий – COMBINE. Слова в новом контейнере будут отсортированы автоматически по ключевому полю, которое мы нигде не меняли, тем самым порядок слов в тексте сохранится.

Пользователю выводится исправленный текст:



**Листинг программы**

#include <iostream>

#include <fstream>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <vector>

#include <ctime>

#include <map>

using namespace std;

class Filtr

{

public:

multimap <int,string> CORRECT; //слова без ошибок

multimap <int,string> NOT\_CORRECT; //слова с ошибками

multimap <int, string> CORRECTED; //исправленные

multimap <int, string> COMBINE; //объединение безошибочных слов с исправленными

void Analization(vector<string> text, int word\_count) // разбиваем на слова с ошибками и без

{

int i= 0;

int j = 0;

int a = 0;

int b = 0;

while (i != word\_count) //перебор всех слов

{

for (j = 0;j < text[i].length();j++) //проверка каждого символа проверяемого слова [i]

{

if (isalpha(text[i][j])) a++;

if (isdigit(text[i][j])) b++;

}

if (a == text[i].length()) CORRECT.insert(pair<int, string>(i,text[i])); //если слово цельное, только из букв - заносим в "без ошибок"

else if (b == text[i].length()) CORRECT.insert(pair<int, string>(i, text[i])); //если число не в слове, а как самостоятельное, то заносим в "без ошибок"

else NOT\_CORRECT.insert(pair<int, string>(i,text[i])); //на корректировку

i++;

a = 0;

b = 0;

}

}

void correction() //исправляем слова в списке "на корректировку"

{

int k = 0;

for (const auto& e : NOT\_CORRECT)

{

string temp;

string temp2;

string temp3;

int i = 0;

int k = 0;

int id = -1;

temp = e.second;

for (i = 0;i < temp.length();i++) //убираем цифры из слова

{

if (!isdigit(temp[i])) //если не цифра записываем в temp2

{

temp2.insert(temp2.end(), temp[i]);

}

}

for (i = 0;i < temp2.length();i++)

{

if (isalpha(temp2[i])) //если буква алфавита записываем в temp3

{

temp3.insert(temp3.end(), temp2[i]); //morn,,,ing

}

else

{

for(k=i;k<temp2.length();k++)

{

if (isalpha(temp2[k])) { i = k-1; id = 0;break; } //если найдет дальше в слове букву, указатель поставит на нее

}

if (id!=0) { temp3.insert(temp3.end(), temp2[i]); break; } //если не нашел букву дальше, то это конец слова, все дальнейшие знаки удаляются, остается только один

id = -1;

}

}

CORRECTED.emplace(e.first, temp3); //добавление в исправленный список

}

}

vector <string> ReWork() //склейка

{

vector <string> forOUTPUT;

int i = 0;

for (const auto& e : CORRECT)

{

COMBINE.emplace(e.first, e.second);

}

for (const auto& e : CORRECTED)

{

COMBINE.emplace(e.first, e.second);

}

for (const auto& e : COMBINE)

{

forOUTPUT.push\_back(e.second);

i++;

}

return forOUTPUT;

}

};

int main(int args, const char \* argv[])

{

setlocale(LC\_ALL, "rus");

string temp;

Filtr Check;

vector<string> slovechki;

ifstream InputTXT("input.txt");

int slovechki\_count = 0;

int CLOCKS\_PER\_MSEC = CLOCKS\_PER\_SEC / 1000; // CLOCKS\_PER\_SEC - количество тактов системных часов в секунду

clock\_t end\_time = clock() + 2\* CLOCKS\_PER\_MSEC; // время завершения 2 ms

while (clock() < end\_time) // цикл ожидания времени

{

InputTXT >> temp;

slovechki.push\_back(temp);

slovechki\_count++;

}

cout <<"Количество слов, считанных за выбранный промежуток времени - "<< slovechki\_count << endl;

cout << "-----------------------------------------------------------------------" << endl;

InputTXT.close();

for (int i = 0;i < slovechki.size();i++)

{

cout << slovechki[i];

cout << " ";

}

cout << endl;

cout << "-----------------------------------------------------------------------" << endl;

Check.Analization(slovechki, slovechki\_count); // разбиваем на слова с ошибками и без

Check.correction(); //исправляем слова с ошибками

slovechki = Check.ReWork(); //склеиваем исправленные слова с нормальными

for (int i = 0;i < slovechki.size();i++)

{

cout << slovechki[i];

cout << " ";

}

cout << endl;

ofstream OutputTXT("output.txt"); //запись в файл

for (int i=0;i<slovechki.size();i++)

{

OutputTXT << slovechki[i];

OutputTXT << " ";

}

OutputTXT.close();

system("pause");

return 0;

}