|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования C++»  Вариант 2 | | |
|  | | Работу выполнил  студент группы ИТ-11-2024 1 курса  Попов Кирилл Дмитриевич  «01» июля 2025 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc202236128)

[Алгоритм решения 3](#_Toc202236129)

[Проверка ввода 3](#_Toc202236130)

[Проверка выполнимости 3](#_Toc202236131)

[Организация слов в список 4](#_Toc202236132)

[Рекурсивный спуск 4](#_Toc202236133)

[Рекурсивный возврат 4](#_Toc202236134)

[Основные функции и методы 5](#_Toc202236135)

[Тестирование 7](#_Toc202236136)

[Некорректные данные 7](#_Toc202236137)

[Корректные данные: 8](#_Toc202236138)

[Нерешаемые задачи 9](#_Toc202236139)

[Код программы 9](#_Toc202236140)

# Постановка задачи

Игра в слова. Дан набор слов. Известно, что эти слова использовались для игры, в которой очередное слово должно заканчиваться той буквой, на которую закончилось предыдущее слово. С какого слова начинается игра, не имеет значения. Если слово оканчивается на «ь», то используется предыдущая буква. Последнее слово цепочки должно заканчиваться буквой первого слова цепочки. Входной файл содержит все слова в одной строке, слова записаны строчными буквами русского алфавита, между словами может быть один или более пробелов. Требуется выстроить данные слова в цепочку согласно правилам игры. Достаточно найти одно подходящее решение. Возможен вариант, что решений не существует.

Пример 1:

(файл входных данных)

хмель мороз налог лимон запах дым лошадь холм гол молох

(файл выходных данных)

налог гол лошадь дым мороз запах холм молох хмель лимон

Пример 2:

(файл входных данных)

аквариум сирень жатва заноза нож авось тормоз мост

(файл выходных данных)

авось сирень нож жатва аквариум мост тормоз заноза

# Алгоритм решения

## Проверка ввода

Все введённые слова должны состоять исключительно из строчных букв русского алфавита, никакое слово не может начинаться с ь, ъ или ы, никакое слово не может заканчиваться двумя мягкими знаками и подобными невозможными комбинациями.

## Проверка выполнимости

Необходимым условием для существования кольца, в котором каждое новое слово начинается на последнюю букву предыдущего является наличие у каждой буквы соответствующей «пары», таким образом для каждой буквы количество слов, начинающихся с неё, должно быть равно количеству слов, заканчивающихся на неё.

## Организация слов в список

Решение подразумевает многократное перемещение групп слов относительно друг друга, двусвязный список будет наиболее оптимальным выбором для хранения слов.

## Рекурсивный спуск

Функция будет перемещать слова, начинающиеся на соответствующие буквы формируя в левой части списка цепь:

**хмель** мороз налог лошадь запах дым лимон холм гол молох

**хмель** **лошадь** мороз налог запах дым лимон холм гол молох

**хмель** **лошадь дым** мороз налог запах лимон холм гол молох

**хмель** **лошадь дым мороз** налог запах лимон холм гол молох

**хмель** **лошадь дым мороз** **запах** налог лимон холм гол молох

**хмель** **лошадь дым мороз** **запах холм** налог лимон гол молох

**хмель** **лошадь дым мороз** **запах холм** **молох** налог лимон гол

В итоге, когда элементов для переноса не останется, в левой части списка образуется кольцо, для оставшихся элементов функция будет вызвана рекурсивно:

хмель лошадь дым мороз запах холм молох **налог** лимон гол

хмель лошадь дым мороз запах холм молох **налог** **гол** лимон

хмель лошадь дым мороз запах холм молох **налог** **гол лимон**

В определённый момент сформируется кольцо, крайний правый элемент которого будет крайним правым элементом списка, с этого момента начнётся рекурсивный возврат.

## Рекурсивный возврат

По завершении функции начнут возвращать итераторы, указывающие на границы колец, получившие итераторы функции попытаются объединить их кольца с собственным кольцом:

хмель лошадь дым мороз запах холм молох **налог гол лимон**

хмель **лимон** **налог гол** лошадь дым мороз запах холм молох

Если некоторые кольца объединить не получилось, функция вернёт их вместе с собственным кольцом. Если на выходе получилось единственное кольцо – ответ найден, если колец несколько – задача не решаема.

## Основные функции и методы

//Метод перемещает элементы ограниченные итраторами аргументами к итератору от которого вызывается

void DList::Iterator::Pull(Iterator Front, Iterator Back) {

const PNode LeftNode = Front.\_Node;

const PNode RightNode = Back.\_Node;

const PNode AnchorNode = this->\_Node;

if (RightNode == AnchorNode || LeftNode == AnchorNode) { //Двигаем узлы в тоже самое место

return;

}

if (RightNode == nullptr) { //Узел "после" крайнего правого

return;

}

bool NodeIsHead = LeftNode->Prev == nullptr;

bool NodeIsTail = RightNode->Next == nullptr;

if (NodeIsHead && NodeIsTail) { //Двигаем все узлы

Front.\_List->\_Head = nullptr;

}

else {

if (NodeIsHead) { //Узлы содержат Head

Front.\_List->\_Head = RightNode->Next;

RightNode->Next->Prev = nullptr;

}

else { //Узлы не содержат Head

LeftNode->Prev->Next = RightNode->Next;

}

if (NodeIsTail) { //Узлы содержат Tail

Front.\_List->\_Tail = LeftNode->Prev;

LeftNode->Prev->Next = nullptr;

}

else { //Узлы не содержат Tail

RightNode->Next->Prev = LeftNode->Prev;

}

}

if (AnchorNode == nullptr) { //Двигаем в самый конец

if (this->\_List->\_Head == nullptr) { //Двигаем в пустой список

this->\_List->\_Head = LeftNode;

LeftNode->Prev = nullptr;

}

else { //Двигаем в конец не пустого спска

this->\_List->\_Tail->Next = LeftNode;

LeftNode->Prev = this->\_List->\_Tail;

}

this->\_List->\_Tail = RightNode;

RightNode->Next = nullptr;

}

else { //Двигаем не в конец

bool AnchorIsHead = AnchorNode->Prev == nullptr;

if (AnchorIsHead) { //Двигаем в начало

AnchorNode->Prev = RightNode;

RightNode->Next = AnchorNode;

this->\_List->\_Head = LeftNode;

LeftNode->Prev = nullptr;

}

else { //Двигаем в середину

AnchorNode->Prev->Next = LeftNode;

LeftNode->Prev = AnchorNode->Prev;

AnchorNode->Prev = RightNode;

RightNode->Next = AnchorNode;

}

}

}

//Пытается из двух петель образовать петлю большего размера, если удалось, возвращает true

bool Merge(Iter Front1, Iter Back1, Iter Front2, Iter Back2) {

Iter Front2AtStart = Front2;

bool MergedSuccessfully = false;

do {

for (Iter IterI = Front1; IterI != Back1+1; ++IterI) {

if ((\*IterI)[0] == (\*Front2)[0]) {

IterI.Pull(Front2, Back2);

MergedSuccessfully = true;

break;

}

}

if (MergedSuccessfully) {

break;

}

if (Front2 != Back2) {

--Back2;

Front2.Pull(Back2 + 1);

--Front2;

}

} while (Front2 != Front2AtStart);

return MergedSuccessfully;

}

//Главная функция, на спуске объеденяет элементы в петли, на возврате объеденяет петли друг с другом

vector<Iter> LoopAndMerge(Iter Front, Iter Back, DList& List) {

Iter IterJ = Front;

Iter IterI = IterJ;

bool LoopIsDone = false;

while (true) {

IterI = IterJ;

++IterJ;

while (true) {

if (IterJ == List.end()) {

LoopIsDone = true;

break;

}

if (LastChar(\*IterI) == (\*IterJ).front()) {

(IterI + 1).Pull(IterJ);

break;

}

++IterJ;

}

if (LoopIsDone) {

break;

}

}

if (IterI == List.Back()) {

return vector<Iter>({Front, IterI});

}

vector<Iter> Loops = LoopAndMerge(IterI + 1, Back, List);

for (auto Loop = Loops.begin(); Loop != Loops.end(); Loop++) {

if (\*Loop != List.end()) {

if (Merge(Front, IterI, \*Loop, \*(Loop+1))) {

\*Loop = \*(Loop + 1) = List.end();

}

}

Loop++;

}

Loops.push\_back(Front);

Loops.push\_back(IterI);

return Loops;

}

# Тестирование

## Некорректные данные

Входные данные:

хмель мороз налог ли1мон запах дым лошадь холм гол молох

Результат:



Входные данные:

хмель мороз налог лимон ьзапах дым лошадь холм гол молох

Результат:



Входные данные:

хмель мороз налог лимон запах дым лошадь холм гол Молох

Результат:



Входные данные:

хмель мороз налог лимон запах дым лошадь холм голъь молох

Результат:



## Корректные данные:

Входные данные:

хмель мороз налог лимон запах дым лошадь холм гол молох

Результат:

хмель лимон налог гол лошадь дым мороз запах холм молох

Входные данные:

аквариум сирень жатва заноза нож авось тормоз мост

Результат:

аквариум мост тормоз заноза авось сирень нож жатва

Входные данные:

аа

Результат:

аа

Входные данные:

аб ба

Результат:

аб ба

## Нерешаемые задачи

Входные данные:

аб ба зг гз

Результат:



# Код программы

https://github.com/Ikromka/IKM2r