|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 1 | | |
| по дисциплине «Структуры данных и алгоритмы» | | |
| **Линейные списки** | | |
|  | | |
|  | Бригада 7 | Дроздов даниил |
| Группа ПМ-15 | иванова дарья |
| Вариант 4 | Чепурная Алина |
|  |  |
|  |  |
|  |  |
| Преподаватель | Тракимус Юрий Викторович |
|  |  |
| Новосибирск, 2022 | | |

1. **Условие задачи**

Задан текст, состоящий из строк, разделенных пробелом и оканчивающийся точкой. Написать подпрограмму поиска заданного элемента в списке. Если в строке текста содержится заданный символ, то переместить его на место последнего символа в этой строке.

1. **Анализ задачи**

Входные данные: искомый символ и текст, состоящий из строк, разделенных пробелом и оканчивающийся точкой.

Выходные данные: список символов, составленный по исходному тексту, в котором все искомые символы перемещены в конец своих строк.

Для решения задачи определяем структуру элемента list, имеющую два поля: символ elem и указатель на следующий элемент next. Сначала объявляем указатель p, указывающий на первый элемент текста и символ target, который будет использоваться в качестве искомого символа. С помощью функции readFile считываем искомый символ и текст из файла в target и p соответственно. Затем с помощью функции moveToStringEnd переместим все вхождения искомого символа на место последнего символа их строк.

Принцип работы функции moveToStringEnd. Принимаемые аргументы – искомый символ t. Объявляем и инициализируем указатели на элементы: seqHead – первый элемент в последовательности искомых символов, seqEnd – последний элемент в последовательности искомых символов, strEnd – конец строки, h – первый элемент во всём тексте, first – первый элемент в текущей строке. Во избежание перемещения первого элемента h на не первое место в последовательности в результате выполнения алгоритма, сделаем фиктивное звено, которое не будет обрабатываться, и поставим его в начало последовательности. Элемент first приравниваем к элементу h. Затем в цикле выполняем следующие действия:

1. Ищем искомый элемент t при помощи функции findElem и присваиваем его к указателю на начало последовательности искомых элементов (seqHead).
2. Затем ищем конец последовательности искомых элементов при помощи функции getSequenceEnd и присваиваем его к указателю на конец последовательности искомых элементов (seqEnd).
3. Ищем конец данной строки с помощью функции getStringEnd и присваиваем его к указателю strEnd.
4. Если конец последовательности искомых элементов совпадает с концом строки, то задача на данной строке считается выполненной. Переходим на следующую строку, присвоив указателю на первый элемент строки указатель на следующую строку (следующий элемент после strEnd). Поиск искомого элемента на следующих итерациях цикла будет начинаться с этого элемента. Иначе переместить последовательность искомых элементов от seqHead до seqEnd в конец strEnd, используя функцию moveSequence.
5. Из цикла выйдем, когда дойдем до конца текста, то есть когда функция поиска искомого элемента findElem вернет NULL.

После выполнения цикла находим первый элемент последовательности с помощью фиктивного звена h и возвращаем его, предварительно удалив фиктивное звено.

Затем остается только вывести получившуюся последовательность. Делаем это при помощи функции print и завершаем работу программы.

1. **Программа**

#include <stdio.h>

const char separator = ' ';

const char endChar = '.';

char target = NULL;

struct list

{

char elem;

list \*next;

list(char \_elem = NULL, list \*\_next = NULL) : elem(\_elem), next(\_next) {}

list \*getStringEnd() // Поиск конца строки

{

if (!this) return NULL;

list \*c = this;

for (; c->next && c->next->elem != separator && c->next->elem != endChar; c = c->next);

return c;

}

list \*findElem(char s) // Поиск первого вхождения элемента s

{

list \*c = this;

for (; c && c->elem != s; c = c->next);

return c;

}

list \*findPrev(list \*h) // Поиск предыдущего элемента

{ // h - первый элемент в списке

list \*c = h;

if (h == this) return NULL;

for (; c && c->next != this; c = c->next);

return c;

}

void moveSequenceTo(list \*sEnd, list \*t, list \*prev)

{ // Перемещение последовательности элементов

if (this == sEnd) // от данного до sEnd на место после t

{ // prev - предыдущий элемент

if (prev) prev->next = next;

next = t->next;

t->next = this;

} else

{

if (prev && sEnd->next) prev->next = sEnd->next;

sEnd->next = t->next;

t->next = this;

}

}

list \*getSequenceEnd() // Поиск конца последовательности одинаковых элементов

{

if (!this) return NULL;

list \*c = this;

for (; c->next && c->next->elem == elem; c = c->next);

return c;

}

list \*moveToStringEnd(char t) // Перемещение всех элементов t в конец строк,

{ // в которых они находятся

if (!next) return this;

list \*h = new list(NULL, this), \*first = h, \*seqHead = this->findElem(t),

\*seqEnd = seqHead->getSequenceEnd(), \*strEnd = seqEnd->getStringEnd();

for (; seqHead;)

{

if (seqEnd != strEnd)

seqHead->moveSequenceTo(seqEnd, strEnd, seqHead->findPrev(first));

else

first = strEnd->next;

seqHead = first->findElem(t);

seqEnd = seqHead->getSequenceEnd();

strEnd = seqEnd->getStringEnd();

}

first = h->next;

delete h;

return first;

}

void output()

{

FILE \*fp;

fopen\_s(&fp, "output.txt", "w");

if (!fp) return;

for (list \*c = this; c; c = c->next)

fputc(c->elem, fp);

fclose(fp);

}

};

list \*input()

{

list \*p = NULL, \*current = NULL;

FILE \*fp;

fopen\_s(&fp, "input.txt", "r");

if (!fp) return NULL;

target = fgetc(fp);

fgetc(fp);

for (char c = fgetc(fp); c != EOF; c = fgetc(fp))

{

if (!current)

{

p = new list();

current = p;

} else

{

current->next = new list();

current = current->next;

}

current->elem = c;

}

fclose(fp);

return p;

}

int main()

{

input()->moveToStringEnd(target)->output();

return 0;

}

1. **Набор тестов**

|  |  |  |
| --- | --- | --- |
| **№** | **Входные данные** | **Назначение** |
| **1** | a  abbbbb. | Искомый символ стоит в начале строки и текста. |
| **2** | a  bbbbbbba. | Искомый символ стоит в конце строки и текста. |
| **3** | a  bbbbabbb. | Искомый символ стоит в середине строки. |
| **4** | a  bbaaabbb. | Искомые символы стоят друг за другом. |
| **5** | a  bbbbbb. | Искомого символа нет в тексте. |
| **6** | a  a. | В тексте нет символов кроме искомого символа. |
| **7** | a  . | Пустой текст. |
| **8** | a  abbbbb abbb abbbb. | Искомый символ стоит в начале строки и текста.  В тексте больше одной строки. |
| **9** | a  bbbabbbb bbababbabbabbb aa. | Искомый символ стоит в середине строки.  В тексте больше одной строки.  В строке несколько непоследовательных искомых символов.  В строке нет символов кроме последовательных искомых символов. |
| **10** | a  abbba aabbbba aaabbbbbaaa. | Искомые символы стоят в начале и в конце строки.  В тексте больше одной строки. |

1. **Результаты работы программы**

|  |  |
| --- | --- |
| **№** | **Ввод/вывод программы** |
| **1** | bbbbba. |
| **2** | bbbbbbba. |
| **3** | bbbbbbba. |
| **4** | bbbbbaaa. |
| **5** | bbbbbb. |
| **6** | a. |
| **7** | . |
| **8** | bbbbba bbba bbbba. |
| **9** | bbbbbbba bbbbbbbbbbaaaa aa. |
| **10** | bbbaa bbbbaaa bbbbbaaaaaa. |