МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

Лабораторная работа №1

по дисциплине «Структуры данных и алгоритмы»

Факультет: ПМИ

Группа: ПМИ-03

Студенты: Сидоров Д.И., Малыгин С. А.

Преподаватель: Еланцева Е.Л.

НОВОСИБИРСК

2021

*1)Условие задачи:* Задан текст, состоящий из строк, разделенных пробелом и оканчивающийся точкой. Написать подпрограмму поиска заданного элемента в списке. Используя эту подпрограмму, подсчитать количество вхождений заданного символа в каждую строку текста

*2)Анализ данных:*

* + *Входные данные:* Последовательность символов (возможно пустая),оканчивающаяся точкой. Заданный символ.
  + *Выходные данные:* Количество вхождений заданной буквы в каждую строку.
  + *Метод решения: Будем посимвольно считывать данные из файла в первый элемент списка и, как только наткнёмся на пробел - перейдём к следующему элементу списка. Будем продолжать так до тех пор, пока не достигнем символа ‘.’ . После чего попросим пользователя ввести искомый символ, количество вхождений которого нужно посчитать. Вызовем процедуру, в которую передадим адрес первого элемента списка, в этот список мы перепишем данные хранящиеся в файле. Далее вызовем процедуру для подсчёта искомого символа, будем идти по списку пока не найдём искомый символ, если такого нет – выведем пользователю сообщение о том что искомого символа в данном списке нет. После чего занесём номер строки и количество найденных элементов в результирующий список и перейдём к следующему элементу списка. А по завершению поиска выведем пользователю значения результирующего списка, а так же запишем эти данные в файл.*
  + *Можно выделить подпрограммы:*

Print\_List– вывод текста.

output\_from\_File - Заполнение списка текстом из файла.

input\_to\_File – запись содержимого списка с результатом выполнения программы в файл.

Count\_of\_Symbol – нахождение количества вхождений заданной буквы в каждую строку.

Delete\_All\_List - удаление списка с текстом.

Delete\_All\_Result - Удаление списка с результатом выполнения программы.

3)Структура входных и выходных данных

Внешнее представление входных данных:

Последовательность символов, оканчивающаяся точкой, в файле Text.txt. Символ, заданный клавиатуры.

Внутреннее представление входных данных:

Линейный однонаправленный ациклический список без заглавного звена. Каждое звено списка реализовано структурой.

struct List{String \*str;List \*next;};

struct String{char symbol; String \*next;};

Внешнее представление выходных данных:

Результат выполнения программы, записанный в файл Text.txt.

Внутреннее представление выходных данных:

Линейный однонаправленный ациклический список без заглавного звена. Каждое звено списка реализовано структурой

struct Result{int №string;int count;Result \*next;};

*4)Алгоритм*

#include<iostream>

#include<fstream>

using namespace std;

Структура String

{

символ symbol;

String \*next;

};

Структура List

{

String \*str;

List \*next;

};

Структура Result

{

Целое число №string;

Целое число count;

Result \*next;

};

Процедура Print\_List(Структура List \*begin\_List)

{

for (List \*flag = begin\_List; flag != NULL; flag = flag->next, Вывод << " ")

{

for (String \*flag\_str = flag->str; flag\_str->next != NULL; flag\_str = flag\_str->next)

{

Вывод << flag\_str->symbol;

}

}

Вывод << endl;

}

Функция output\_from\_File(Структура List \*begin\_List)

{

List \*flag\_List = begin\_List;

String \*flag\_String = begin\_List->str;

ifstream fin;

Открыть файл("Text.txt");

Если (!fin.is\_open())

{

Вывод << "Ошибка открытия файла!" << endl;

Возвращаем false;

}

Иначе

{

for (символ ch = NULL; ch != '.';)

{

fin.get(ch);

Если (ch != ' ')

{

flag\_String->symbol = ch;

flag\_String->next = new String;

flag\_String = flag\_String->next;

flag\_String->next = NULL;

}

Иначе

{

flag\_List->next = new List;

flag\_List = flag\_List->next;

flag\_List->next = NULL;

flag\_List->str = new String;

flag\_String = flag\_List->str;

flag\_String->next = NULL;

}

}

}

Закрыть файл();

Возвращаем true;

}

Функция input\_to\_File(Структура Result \*begin\_Result, Символ ch)

{

ofstream fout;

Открыть файл для записи("Text.txt", ios\_base::app);

Если (!fout.is\_open())

{

Вывод << "Ошибка открытия файла!" << endl;

Возвращаем false;

}

Иначе

{

Вывод << endl << "Символ " << ch;

Запись в файл << endl << "Символ " << ch;

for (; begin\_Result->next != NULL; begin\_Result = begin\_Result->next)

{

Вывод << endl << "В строке " << begin\_Result->№string << " - " << begin\_Result->count << " символов.";

Запись в файл << endl << "В строке " << begin\_Result->№string << " - " << begin\_Result->count << " символов.";

}

Закрыть файл для записи();

}

Возврашаем true;

}

Процедура Count\_of\_Symbol(Структура List \*begin\_List, Структура Result \*begin\_Result, Символ ch)

{

Result \*flag\_Result = begin\_Result;

Целое число count(0);

Целое число i(1);

Целое число Num(0);

for (List \*flag = begin\_List; flag != NULL; flag = flag->next, i++)

{

for (String \*flag\_str = flag->str; flag\_str->next != NULL; flag\_str = flag\_str->next)

{

Если (flag\_str->symbol == ch)

{

count++;

}

}

Если (count != 0)

{

flag\_Result->№string = i;

flag\_Result->count = count;

flag\_Result->next = new Result;

flag\_Result = flag\_Result->next;

flag\_Result->next = NULL;

Num += count;

count = 0;

}

}

Если (Num > 0)

{

input\_to\_File(begin\_Result, ch);

}

Иначе

{

Вывод << "Искомый символ не найден!" << endl;

}

}

Процедура Delete\_All\_List(struct List \*begin)

{

List \*f = begin->next;

List \*fg = begin;

String \*f\_str = fg->str->next;

String \*fg\_str = fg->str;

Пока (f->next != NULL)

{

Пока (f\_str->next != NULL)

{

Удалить fg\_str;

fg\_str = f\_str;

f\_str = f\_str->next;

}

Удалить f\_str;

Удалить fg;

fg = f;

f = f->next;

f\_str = fg->str->next;

fg\_str = fg->str;

}

Удалить f;

}

Процедура Delete\_All\_Result(Структура Result \*begin)

{

Result \*f = begin->next;

Result \*fg = begin;

Пока (f->next != NULL)

{

Удалить fg;

fg = f;

f = f->next;

}

Удалить f;

}

Главная функция

{

Подключение русского языка;

List \*begin\_List = new List;

begin\_List->next = NULL;

begin\_List->str = new String;

begin\_List->str->next = NULL;

Функция output\_from\_File(begin\_List);

Процедура Print\_List(begin\_List);

Символ ch;

Ввод >> ch;

Вывод << endl;

Result \*begin\_Result = new Result;

begin\_Result->next = NULL;

Процедура Count\_of\_Symbol(begin\_List, begin\_Result, ch);

Процедура Delete\_All\_List(begin\_List);

Процедура Delete\_All\_Result(begin\_Result);

Возвращаем 0;

}

*5)Структура программы:*

1) Процедура вывода текста:

Print\_List(struct List \*begin\_List)

Входные данные: \*begin\_List – указатель на начало списка с текстом.

2)Функция заполнения списка текстом из файла:

output\_from\_File(struct List \*begin\_List)

Входные данные: \*begin\_List – указатель на начало списка с текстом.

Выходные данные: True or false.

3)Функция записи содержимого списка с результатом выполнения программы в файл:

input\_to\_File(struct Result \*begin\_Result, char ch)

Входные данные: \*begin\_Result – указатель на начало списка с текстом.

Ch - заданный символ.

Выходные данные: True or false.

4) Процедура нахождения количества вхождений заданной буквы в каждую строку:

Count\_of\_Symbol(struct List \*begin\_List, struct Result \*begin\_Result, char ch)

Входные данные: \*begin\_List – указатель на начало списка.

\*begin\_Result - указатель на начало списка с результатом выполнения программы.

Ch- заданный символ.

5) Процедура удаления списка с текстом:

Delete\_All\_List(struct List \*begin)

Входные данные: \*begin – указатель на начало списка

6) Процедура удаления списка с результатом выполнения программы:

Delete\_All\_Result(struct Result \*begin)

Входные данные: \*begin – указатель на начало списка

Delete\_All\_List

Print\_List

main

Delete\_All\_Result

Count\_of\_Symbol

output\_from\_File

input\_to\_File

*6)Текст программы:*

#include<iostream>

#include<fstream>

using namespace std;

struct String

{

char symbol;

String \*next;

};

struct List

{

String \*str;

List \*next;

};

struct Result

{

int №string;

int count;

Result \*next;

};

void Print\_List(struct List \*begin\_List)

{

for (List \*flag = begin\_List; flag != NULL; flag = flag->next, cout << " ")

{

for (String \*flag\_str = flag->str; flag\_str->next != NULL; flag\_str = flag\_str->next)

{

cout << flag\_str->symbol;

}

}

cout << endl;

}

bool output\_from\_File(struct List \*begin\_List)

{

List \*flag\_List = begin\_List;

String \*flag\_String = begin\_List->str;

ifstream fin;

fin.open("Text.txt");

if (!fin.is\_open())

{

cout << "Ошибка открытия файла!" << endl;

return false;

}

else

{

for (char ch = NULL; ch != '.';)

{

fin.get(ch);

if (ch != ' ')

{

flag\_String->symbol = ch;

flag\_String->next = new String;

flag\_String = flag\_String->next;

flag\_String->next = NULL;

}

else

{

flag\_List->next = new List;

flag\_List = flag\_List->next;

flag\_List->next = NULL;

flag\_List->str = new String;

flag\_String = flag\_List->str;

flag\_String->next = NULL;

}

}

}

fin.close();

return true;

}

bool input\_to\_File(struct Result \*begin\_Result, char ch)

{

ofstream fout;

fout.open("Text.txt", ios\_base::app);

if (!fout.is\_open())

{

cout << "Ошибка открытия файла!" << endl;

return false;

}

else

{

cout << endl << "Символ " << ch;

fout << endl << "Символ " << ch;

for (; begin\_Result->next != NULL; begin\_Result = begin\_Result->next)

{

cout << endl << "В строке " << begin\_Result->№string << " - " << begin\_Result->count << " символов.";

fout << endl << "В строке " << begin\_Result->№string << " - " << begin\_Result->count << " символов.";

}

fout.close();

}

return true;

}

void Count\_of\_Symbol(struct List \*begin\_List, struct Result \*begin\_Result, char ch)

{

Result \*flag\_Result = begin\_Result;

int count(0);

int i(1);

int Num(0);

for (List \*flag = begin\_List; flag != NULL; flag = flag->next, i++)

{

for (String \*flag\_str = flag->str; flag\_str->next != NULL; flag\_str = flag\_str->next)

{

if (flag\_str->symbol == ch)

{

count++;

}

}

if (count != 0)

{

flag\_Result->№string = i;

flag\_Result->count = count;

flag\_Result->next = new Result;

flag\_Result = flag\_Result->next;

flag\_Result->next = NULL;

Num += count;

count = 0;

}

}

if (Num > 0)

{

input\_to\_File(begin\_Result, ch);

}

else

{

cout << "Искомый символ не найден!" << endl;

}

}

void Delete\_All\_List(struct List \*begin)

{

List \*f = begin->next;

List \*fg = begin;

String \*f\_str = fg->str->next;

String \*fg\_str = fg->str;

while (f->next != NULL)

{

while (f\_str->next != NULL)

{

delete fg\_str;

fg\_str = f\_str;

f\_str = f\_str->next;

}

delete f\_str;

delete fg;

fg = f;

f = f->next;

f\_str = fg->str->next;

fg\_str = fg->str;

}

delete f;

}

void Delete\_All\_Result(struct Result \*begin)

{

Result \*f = begin->next;

Result \*fg = begin;

while (f->next != NULL)

{

delete fg;

fg = f;

f = f->next;

}

delete f;

}

int main()

{

setlocale(LC\_ALL, "rus");

List \*begin\_List = new List;

begin\_List->next = NULL;

begin\_List->str = new String;

begin\_List->str->next = NULL;

output\_from\_File(begin\_List);

Print\_List(begin\_List);

char ch;

cin >> ch;

cout << endl;

Result \*begin\_Result = new Result;

begin\_Result->next = NULL;

Count\_of\_Symbol(begin\_List, begin\_Result, ch);

Delete\_All\_List(begin\_List);

Delete\_All\_Result(begin\_Result);

return 0;

}

*7)Тесты:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Входные данные* | | *Выходные данные Text.txt* | *Примечание* |
| *Text.txt* | *Заданный символ* |
| *1* | Hello my world. | *l* | Символ l  В строке 1 - 2 символов.  В строке 3 - 1 символов. | *Программа работает правильно* |
| *2* | I go to home. | *o* | Символ o  В строке 2 - 1 символов.  В строке 3 - 1 символов.  В строке 4 - 1 символов. | *Программа работает правильно* |
| *3* | Hello my world. | *g* |  | *Искомый символ не найден!* |
| *4* |  |  |  | *Программа не выполняется* |
| *5* | Ext.txt |  |  | *Ошибка открытия файла* |
| *6* | Hello my world. | *.* | Символ .  В строке 3 - 1 символов. | *Программа работает правильно* |

8) *Результат работы программы*:

Программа работает правильно, что подтверждают тесты.