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();

	}
```

```
	Возвращаем 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.

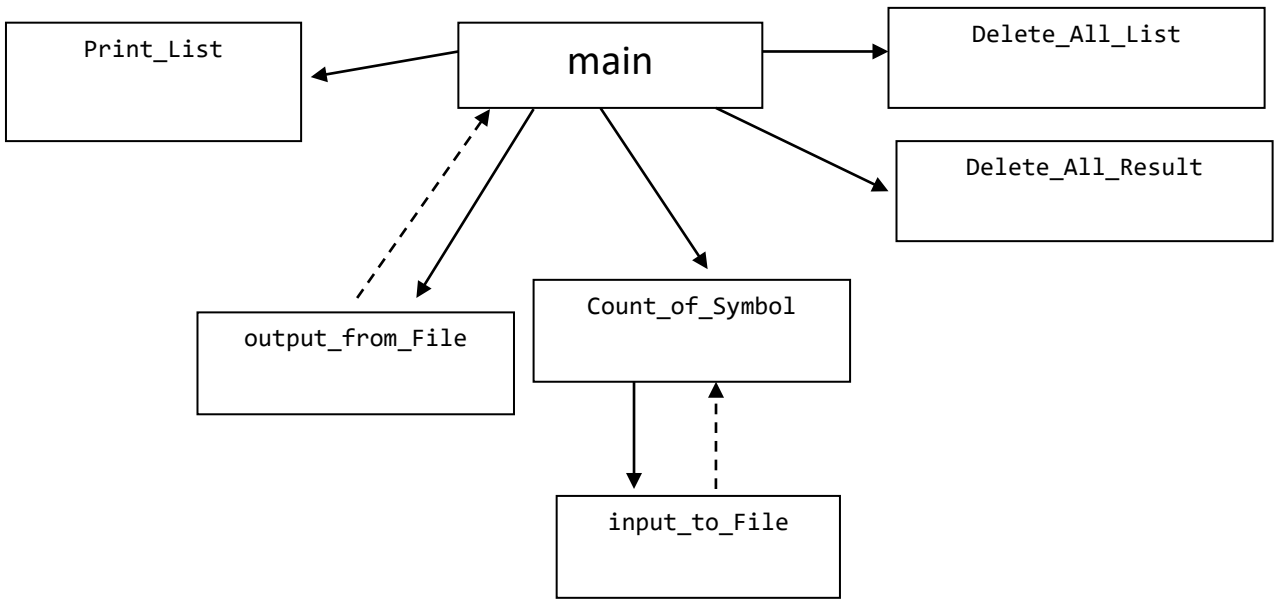
4) Процедура нахождения количества вхождений заданной буквы в каждую строку:
```
Count_of_Symbol(struct List *begin_List, struct Result *begin_Result, char ch)
```
Входные данные: `*begin_List` – указатель на начало списка.

`*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



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.