Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

СПИСКИ

Отчет по лабораторной работе №3

По дисциплине

«Структуры и алгоритмы обработки данных в ЭВМ»

Студент гр. 431-3

_____ А.В. Гурулёв

«___»_____ 2022 г.

Проверил: профессор кафедры АСУ, д.т.н.

_____ А.Н. Горитов

«___»_____ 2022 г.

Томск 2022

1 Задание на лабораторную работу

Подготовить текстовый файл, содержащий не менее 10 слов. Прочитать данные из этого файла и сформировать двусвязный список, элементы которого имеют тип STRING. Написать программу, подсчитывающую количество элементов списка, которые начинаются с того же символа, что и следующий элемент списка.

После завершения работы со списками освободите занимаемую ими динамическую память.

2 Алгоритм решения задачи

1. Открываем файл;
2. Организуем цикл с условием: “Пока возможно считывание слова”. Тело цикла:
3. Заносим считанное слово в список;
4. Конец цикла;
5. Считываем начало списка в переменную x;
6. Организуем цикл с условием: “Пока x не равен концу списка”. Тело цикла:
7. Если первый символ поля info равен первому символу поля info следующего узла то:
8. Увеличиваем счётчик;
9. Конец если;
10. Заносим в x следующий узел;
11. Конец цикла;
12. Выводим результат счётчика;
13. Чистим память

3 Листинг программы

Для main.cpp:

```
#include <iostream>
#include <locale.h>
#include <fstream>

#include "ListLogic.cpp"
```

```
using namespace std;
using namespace myTask;


int main()
{
  setlocale(LC_ALL, "rus");

  MyList list;

  ifstream f("File.txt");

  if (f.is_open())
  {
        string str;
        while (f >> str)
        {
              list.insertEnd(str);
        }
        Node* x = list.head;
        int count = 0;
        while (x!=list.tail)
        {
              if ((*x).info[0] == (*(*x).next).info[0])
              {
                    count++;
              }
              x = (*x).next;
        }
        cout << count << endl;
```

```
  }
  else
  {
        cout << "error" << endl;
  }

  system("pause");

  return 0;
}
```

Для ListLogic.cpp:

```
#include <iostream>
#include <string>
using namespace std;

namespace myTask
{
  struct Node
  {
        string info;
        Node* next;
        Node* last;
  };

  class MyList
  {
  public:
        //Голова и хвост
        Node* head;
        Node* tail;

        //Инициализация
        MyList()
```

```
{
	head = NULL;
	tail = NULL;
}

~MyList()
{
	Node* x = head;
	while (x)
	{
		Node* next = (*x).next;
		(*x).info = "";
		(*x).last = NULL;
		(*x).next = NULL;
		x = next;			}
}

//Функция вставки перед узлом
void insertBefore(Node* node, string newData)
{
	Node* newNode = new Node;
	(*newNode).info = newData;
	(*newNode).next = node;
	if ((*node).last)
	{
		(*newNode).last = (*node).last;
		(*(*node).last).next = newNode;
	}
	else
	{
		(*newNode).last = NULL;
		head = newNode;
	}
	(*node).last = newNode;
}
```

```
//Функция вставки после узла
void insertAfter(Node* node, string newData)
{
	Node* newNode = new Node;
	(*newNode).info = newData;
	(*newNode).last = node;
	if ((*node).next)
	{
		(*newNode).next = (*node).next;
		(*(*node).next).last = newNode;
	}
	else
	{
		(*newNode).next = NULL;
		tail = newNode;
	}
	(*node).next = newNode;
}

//Функция вставки узла в начало, возможно пустого списка
void insertBeginning(string newData)
{
	if (head)
	{
		insertBefore(head, newData);
	}
	else
	{
		head = new Node;
		(*head).info = newData;
		(*head).next = NULL;
		(*head).last = NULL;
		tail = head;
	}
```

```
}

//Функция вставки узла в конец
void insertEnd(string newData)
{
	if (tail)
	{
		insertAfter(tail, newData);
	}
	else
	{
		insertBeginning(newData);
	}
}

//Функция поиска(с начала) элемента из списка
Node* findAtStart(string data)
{
	Node* answer = head;

	while (answer)
	{
		if ((*answer).info == data)
		{
			return answer;
		}

		answer = (*answer).next;
	}

	cout << "Элемент не найден";
	exit(EXIT_FAILURE);
}

//Функция поиска(с конца) элемента из списка
```

```
Node* findAtEnd(string data)
{
	Node* answer = tail;

	while (answer)
	{
		if ((*answer).info == data)
		{
			return answer;
		}

		answer = (*answer).last;
	}

	cout << "Элемент не найден";
	exit(EXIT_FAILURE);
}

//Функция взятия из списка
string remove(string data)
{
	Node* node = findAtStart(data);

	if (node == head)
	{
		if ((*node).next)
		{
			head = (*node).next;
			(*head).last = NULL;
			string out = (*node).info;
			delete[] node;
			return out;
		}
		else
		{
```

```
                head = NULL;
                tail = NULL;
                string out = (*node).info;
                delete[] node;
                return out;
        }
}
else
{
        if (node == tail)
        {
                if ((*node).last)
                {
                        tail = (*node).last;
                        (*tail).next = NULL;
                        string out = (*node).info;
                        delete[] node;
                        return out;
                }
                else
                {
                        head = NULL;
                        tail = NULL;
                        string out = (*node).info;
                        delete[] node;
                        return out;
                }
        }
        else
        {
                (*(*node).last).next = (*node).next;
                (*(*node).next).last = (*node).last;
                string out = (*node).info;
                delete[] node;
                return out;
```

```
                }
            }
        }

        //Функция взятия из списка (первого)
        string remove()
        {
            Node* node = head;
            if (node)
            {

                if ((*node).next)
                {
                    head = (*node).next;
                    (*head).last = NULL;
                    string out = (*node).info;
                    delete[] node;
                    return out;
                }
                else
                {
                    head = NULL;
                    tail = NULL;
                    string out = (*node).info;
                    delete[] node;
                    return out;
                }
            }
            else
            {
                return NULL;
            }
        }
    };
}
```

4 Пример решения

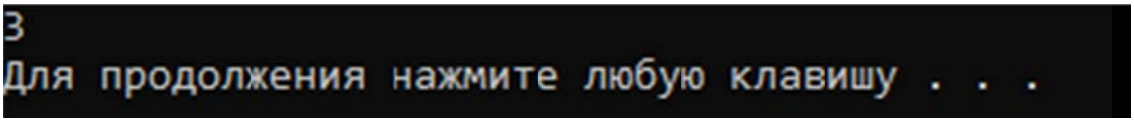

Рисунок 4.1 - Результат выполнения программы

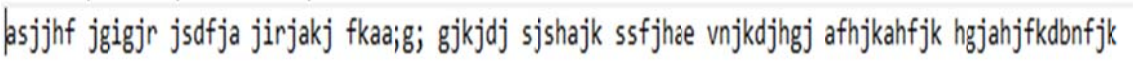

Рисунок 4.2 - Входные данные

5 Вывод

Я изучил принципы работы со структурой данных список и его преимущество перед простыми массивами.