МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсивная обработка иерархических списков

Студент гр. 9384 _____ Гурин С.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить способ реализации структуры иерархического списка и принцип его обработки.

Задание.

ВАРИАНТ 3.

Заменить в иерархическом списке все вхождения заданного элемента (атома) x на заданный элемент (атом) y.

Анализ задачи.

Задача заключается в том, что нужно реализовать структуру иерархического списка, создать функции его заполнения. Далее следует реализовать функцию замены атомов x на атом y.

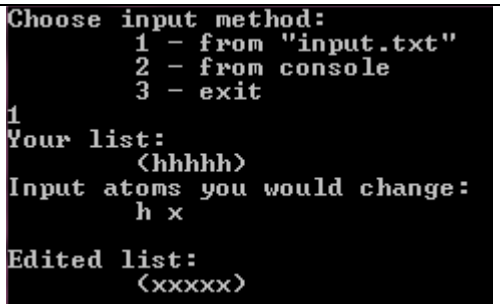
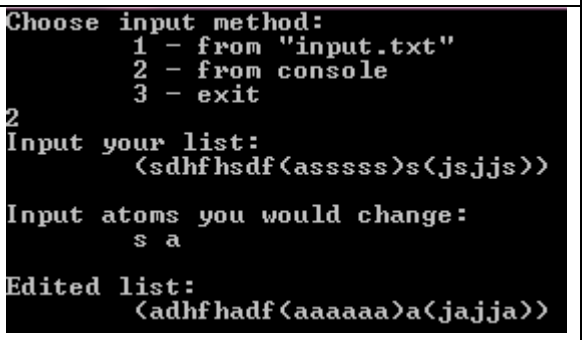
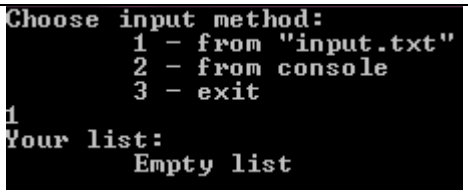
Выполнение работы.

При запуске программы пользователю предлагается выбрать способ ввода списка: с файла “input.txt” или с консоли. Затем пользователь вводит символы (атомы), которые он хочет поменять.

Далее выполняется функция `void change(list s, char x, char y)`, принимающая сам список, который нужно изменить, значение x (атом, который нужно заменить) и значение y (атом, на который нужно заменить x).

Функция `change()` сначала проверяет не пустой ли список, если это так, то идет проверка не является ли данный список атомов. Если не является, то рекурсивно вызывается функция `change()` обрабатывающая голову списка, а так же хвост, если он есть. В противном случае, если данный список является атомом, то происходит проверка на то, что является ли данный атом искомым элементом. Если является, то происходит замена.

Тестирование.

№ теста	ввод		вывод
	console	input.txt	
1	1 h x	(hhhhhh)	
2	2 (sdhfhsdf(asssss)s(jsjjs)) s a		
3	1	()	

Выводы.

При выполнении данной лабораторной работы был изучен принцип создания структуры иерархического списка и принцип работы с ним с помощью рекурсивных функций. При выполнении этой работы применялись знания о работе с файлами и знания о рекурсивных функциях.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include "hier_list.h"

using namespace std;
using namespace H_list;


int main()
{
	list s;
	char ch;
	char x, y;
	cout << "Choose input method:\n\t1 - from \"input.txt\"\n\t2 - from
console\n\t3 - exit\n";
	cin >> ch;
	switch (ch){
	case '1':
		file_readlist(s);
		cout << "Your list:\n\t";
		write_list(s);
		break;
	case '2':
		cout << "Input your list:\n\t";
		read_list(s);
		break;
	case '3':
		return 0;
	default:
		cout << "Wrong argument\n";
		return 0;
	}
	cout << "\nInput atoms you would change:\n\t";
	cin >> x >> y;
	change(s, x, y);
```

```
	cout << "\nEdited list:\n\t";
	write_list(s);
	cout << "\n";
	destroy(s);
	return 0;
}
```

Название файла: hier_list.cpp

```
#include <iostream>
#include <fstream>
#include "hier_list.h"

using namespace std;

namespace H_list {
	list head(const list s) {
		if (s != nullptr)
			if (!isAtom(s))
				return s->node.pair.hd;
			else {
				cout << "Error:Head(atom)\n";
				exit(1);
			}
		else {
			cout << "Error:Head(nil)\n";
			exit(1);
		}
	}
	list tail(const list s) {
		if (s != nullptr) {
			if (s != nullptr)
				if (!isAtom(s)) return s->node.pair.tl;
				else {
					cout << "Error: Tail(atom)\n";
					exit(1);
				}
			else {
				cout << "Error: Tail(nil)\n";
				exit(1);
```

```
            }
        }
}
bool isAtom(const list s) {
        if (s == nullptr)
            return false;
        else
            return (s->tag);
        }
bool isNull(const list s) {
        return s == nullptr;
}
list cons(const list h, const list t) {
        list p;
        if (isAtom(t)) {
            cout << "Error: cons(*,atom)\n";
            exit(1);
        }
        else {
            p = new s_expr;
            if (p == nullptr) {
                cout << "Memory overloaded\n";
                exit(1);
            }
            p->tag = false;
            p->node.pair.hd = h;
            p->node.pair.tl = t;
            return p;
        }
}
list make_atom(const base x) {
        list s;
        s = new s_expr;
        s->tag = true;
        s->node.atom = x;
        return s;
}
void destroy(list s){
```

```
      if (s != nullptr) {
            if (!isAtom(s)) {
                  destroy(head(s));
                  destroy(tail(s));
            }
            delete s;
      }
}
base getAtom(const list s) {
      if (!isAtom(s)) {
            cout << "Error: getAtom(s) for !isAtom(s) \n";
            exit(1);
      }
      else
            return s->node.atom;
}
void read_list(list& y) {
      base x;
      do
            cin >> x;
      while (x == ' ');
      read_s_expr(x, y);
}
void read_s_expr(base prev, list& y) {
      if (prev == ')') {
            cout << "! List.Error 1\n";
            exit(1);
      }
      else
            if (prev != '('
                  y = make_atom(prev);
            else read_seq(y);
}
void read_seq(list& y) {
      base x;
      list p1, p2;
      if (!(cin >> x)) {
            cout << "! List.Error 2\n";
```

```
            exit(2);
        }
        else {
            while (x == ' ')
                cin >> x;
            if (x == ')')
                y = nullptr;
            else{
                read_s_expr(x, p1);
                read_seq(p2);
                y = cons(p1, p2);
            }
        }
}
void write_list(const list x) {
        if (isNull(x)) {
            cout << "Empty list\n";
            exit(1);
        }
        else
            if (isAtom(x))
                cout << x->node.atom;
            else {
                cout << '(';
                write_seq(x);
                cout << ')';
            }
}
void write_seq(const list x) {
        if (!isNull(x)) {
            write_list(head(x));
            write_seq(tail(x));
        }
}
void file_readlist(list& y) {
        string str;
        ifstream file;
        base x;
```

```
	file.open("input.txt");
	if (!file.is_open()) {
		cout << "Error opening file (input.txt)\n";
		exit(0);
	}
	do
		file >> x;
	while (x == ' ');
	file_read_s_expr(x, y, file);
	file.close();
}
void file_read_s_expr(base prev, list& y, ifstream& file) {
	if (prev == ')') {
		cout << "! List.Error 1\n";
		exit(1);
	}
	else
		if (prev != '(')
			y = make_atom(prev);
		else file_read_seq(y,file);
}
void file_read_seq(list& y, ifstream& file) {
	base x;
	list p1, p2;
	if (!(file >> x)) {
		cout << "! List.Error 2\n";
		exit(2);
	}
	else {
		while (x == ' ')
			file >> x;
		if (x == ')')
			y = nullptr;
		else {
			file_read_s_expr(x, p1, file);
			file_read_seq(p2, file);
			y = cons(p1, p2);
		}
```

```
            }
      }
      void change(list& s, char x, char y) {
            if (!isNull(s)) {
                  if (!isAtom(s)) {
                        change(s->node.pair.hd, x, y);
                        if (s->node.pair.tl) {
                              change(s->node.pair.tl, x, y);
                        }
                  }
                  else {
                        if (s->node.atom == x) {
                              s->node.atom = y;
                        }
                  }
            }
      }
}
```

Название файла: hier_list.h

```
#include <iostream>
#include <fstream>

using namespace std;

namespace H_list {
      typedef char base;
      struct s_expr;
      struct two_ptr {
            s_expr* hd;
            s_expr* tl;
      };
      struct s_expr {
            bool tag;
            union {
                  base atom;
                  two_ptr pair;
            }node;
      };
```

```
    typedef s_expr* list;

    base getAtom(const list s);
    list head(const list s);
    list tail(const list s);
    list cons(const list h, const list t);
    bool isAtom(const list s);
    bool isNull(const list s);
    void destroy(list s);

    void read_list(list& y);
    void read_s_expr(base prev, list& y);
    void read_seq(list& y);

    void write_list(const list x);
    void write_seq(const list x);

    void file_readlist(list& y);
    void file_read_s_expr(base prev, list& y, ifstream& file);
    void file_read_seq(list& y, ifstream& file);

    void change(list& y, char atom1, char atom2);
}
```

Название файла: Makefile

```
all:
    g++ hier_list.cpp main.cpp && clear && ./a.out
```