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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: ИЕРАРХИЧЕСКИЕ СПИСКИ

Студент гр. 7382 _____ Гиззатов А.С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2018

Задание.

Варинт №14.

Обратить иерархический список на всех уровнях вложения; например, для исходного списка (a (b c) d) результатом обращения будет список (d (c b) a).

Функции и структуры данных

Были написаны **5** функции и 1 класс**:1)**Функция обращающая список в новый список начинающийся с хвостаList *rev(List *head)**,2)**Функция находящая конец вложенного списка int srch_sub_list(std::string str)**,3)**Функция,которая проверяет скобки int br_check(std::string str) ,4)Функция проверяющая строку на годность к использованию bool is_valuable(std::string str),5) Функция main.

Описание алгоритма:

1)В функции main считывается строка,потом из этой строки удаляется все пробелыю

2)Строка проверяется на правильность.

3)Создается список и выписывается на экран для уверенности что он создался правильно.

4)Список реверсируется.

5)Новый список выписывается на экран

Тестирование.

Были написаны 4 теста для данной программы, а также скрипт для тестирования и компиляции программы. Рассмотрим на примере конкретного теста работу программы. Возьмем 1 тест и после выполнения функции реверсирования она будет выглядеть (h(gf)dcba). Функция rev меняет указатели на элементы списка таким образом, что голова становится хвостом,а хвост головой.

№	Входные данные	Выходные данные
1	(abcd(fg)h)	(h(gf)dcba)
2	(abcd))	text is not correct

3	(abcd)	(dcba)
4	(abcd(mn(jk))k)	(k((kj)nm)dcba)

Выводы.

В результате работы были усвоены методы создания иерархического списка, а также написана программа с использованием класса и его методов.

Исходный код

```
#include <iostream>
#include <cctype>
#include <string>
#include <regex>
int srch_sub_list(std::string str);
class List{
	public:
	char El;
	List *next;
	List *Sub_list;
	List(std::string str){
		if(str.length() != 0){
			if(str[0] =='('){
				str.erase(0,1);
				str.erase(str.length()-1,1);
			}
			El = str[0];
			if(str[1]=='('){
				//str.erase(0,1);
				std::string sub_str = str.substr(1,srch_sub_list(str));
				//std::cout<<str.substr(1,srch_sub_list(str))<<" "<<
srch_sub_list(str) <<std::endl;
				Sub_list = new List(sub_str);
				str.erase(1,sub_str.length());
```

```
            }
            str.erase(0,1);
            next = new List(str);
      }
      else{
            El = '\0';
            next = nullptr;
            Sub_list = nullptr;
      }

}
~List(){
      delete next;
      delete Sub_list;
}
List(char s,List *sub_str_list){
      El = s;
      Sub_list = sub_str_list;
}
int count(){
      int res=0;
      List *ptr= next;
      while(ptr !=nullptr){
            if(ptr!=nullptr){
                  res++;
            }
            ptr=ptr->next;
      }
      return res;
}

void print_list(){
      if(Sub_list != nullptr){
            std::cout<< "(";
                  Sub_list->print_list();
                  std::cout<<")";
               std::cout <<  El;
          }
      else{
            std::cout <<  El;
      }
```

```
            if(next != nullptr){
                  next->print_list();
            }
      }
};
List *rev(List *head) {
      List *ptr1 = head;
      List *ptr2;
      List *ptr_last;
      List *rev_list;
      while (ptr1->next != nullptr) {
            if(ptr1->Sub_list != nullptr){
                  ptr1->Sub_list=rev(ptr1->Sub_list);
            }
            rev_list = ptr1;
            ptr1 = ptr1->next;

      }
      ptr1 = head->next;
      head->next = nullptr;
      ptr2 = head;
      head = ptr1;
      ptr1 = ptr1->next;
      while (ptr1!= nullptr) {
            head->next = ptr2;
            ptr2 = head;
            std::cout << "head " << ptr2->El <<std::endl;
            head = ptr1;
            ptr1 = ptr1->next;
      }
      return rev_list;
}

int srch_sub_list(std::string str){
      int i = 0 ,open_br=0,k,close_br = 0;
            do{
                  if(str[i] == ')'){
                        k = i;
                  }
                  i++;
            }
      while(i<str.length());
```

```
            return k;
}

int br_check(std::string str){
      int i=0,open_br=0,close_br=0;
      for(i;i<str.length();i++){
                        if(str[i]=='('){
                                open_br++;
                        }
                        if(str[i] == ')'){
                                close_br++;
                       }
      }
        if(open_br==close_br)
                return i;
        else
                return -1;

}

bool is_valuable(std::string str){
      if(br_check(str)>0){
            for(int i=0;i<str.length()-1;i++){
                  if(str[i]=='(' && str[i+1]=='('){
                        return false;
                  }
                  if(str[i] == ')' && str[i+1] == '('){
                        return false;
                  }
            }
            return true;
      }
      else{
            return false;
      }

}

int main(){
      std::string str;
      getline(std::cin,str);
      str=std::regex_replace(str,std::regex(" "),"");
```

```
	std::cout << str << std::endl;
	if(is_valuable(str)){
		List *lst = new List(str);
		lst->print_list();
		std::cout<<std::endl;
		List *rev_list = rev(lst);
		std::cout << "(";
		rev_list->print_list();
		std::cout<<")"<<std::endl;

	}
	else{
		std::cout << "text is not correct" << std::endl;
		return 0;
	}
	return 0;
}
```