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 _____ **Государкин Я. С.**

Преподаватель _____ **Фирсов М. А.**

Санкт-Петербург
2018

Цель работы.

Ознакомиться с основными методами обработки списков.

Основные теоретические положения.

Реку́рсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.

Список — структура данных, позволяющая эффективно работать с большими данными за счет дешевых операций вставки, удаления, и т.д.

Выполнение работы.

Проверить структурную идентичность двух иерархических списков(списки структурно идентичны, если их устройство (скобочная структура и количество элементов в соответствующих (под)списках) одинаково, при этом атомы могут отличаться);

В работе используется язык программирования C++;

исходный код: файл main.cpp

скрипты: launcher, userlauncher

Сначала были написаны необходимые для успешного выполнения задания функции: Brackets(string &str); BracketRework(string &str);

Brackets(string &str) — функция принимающая на вход строку по ссылке, которая проходит проверку на то, является ли она вообще иерархическим списком, т.е если последовательность скобок нарушена, или на вход поданы какие-то левые данные, например числа до первой открывающей скобки, или буквы после последней закрывающей скобки, то функция выведет false. Проверка реализована с помощью библиотечного стека.

BracketRework(string &str) — функция принимающая на вход строку по ссылке. Обеспечивает «очищение» строки от атомов, т.е от символов, строк, чисел, и прочего, не имеющего общего со СТРУКТУРОЙ списка. Не имеет возвращающего значения.

После подготовки пишется main.
Из cin выгружаются строки, проходят проверку на свою «списочность», если проходят, то каждая строка очищается от атомов и сравниваются только структуры строк. При любом раскладе программа выведет результат сравнения.

Также есть предварительный вывод строк, и небольшой скрипт, который позволяет загружать в программу сразу много тестов из одного файла и выводит результаты в терминал/ком. строку.

Тестирование программы:

Работа была протестирована практически всеми возможными способами. В таблице ниже — результаты.

Входные данные	Результат работы программы
(g(f(d()a)d)f) (gregeg(grthrty(4u5ui6k([gh]))))	String 1 && 2 are structure identical
(()))(	[ERROR] One of strings isn't brackets. Aborting!
()gthr ((grgeh)hr))	[ERROR] One of strings isn't brackets. Aborting!
qffw 124142	[ERROR] One of strings isn't brackets. Aborting!
(dqwd()fewf) ()()	[ERROR] One of strings isn't brackets. Aborting!
(dqwd()fe(fg(a)are)) (gewrer(fgwe)(()))	String 1 && 2 are structure identical
123(vodka_javit(dlya{}{}(tebya))) (ilusornost'(b(y(t(i(y(a)))))	[ERROR] One of strings isn't brackets. Aborting!
(noo(you)___(are)(bracket)(master)) (()()()())	String 1 && 2 are structure identical
123(rthrt()hrtj()(gre())g) (rtehrt9(perebzet_sig_stop))	[ERROR] One of strings isn't brackets. Aborting!
(hren(tebe)na(vorotnik)) (bez(moz)gla(ya))	String 1 && 2 are structure identical

Выводы.

В ходе выполнения лабораторной работы получены знания по теме «иерархические списки» и закреплены знания синтаксиса языка C++;

ИСХОДНЫЙ КОД:

```
#INCLUDE <VECTOR>
#INCLUDE <IOSTREAM>
#INCLUDE <STACK>
#INCLUDE <CSTDLIB>

USING NAMESPACE STD;

BOOL BRACKETS(STRING &STR);
BOOL EQUAL(STRING &STR1, STRING &STR2);
VOID BRACKETREWORK(STRING &STR);

INT MAIN() {
   STRING STR1;
   STRING STR2;
   COUT << "ENTER 1-ST STRING" << ENDL;
   CIN >> STR1;
   COUT << "ENTER 2-ND STRING" << ENDL;
   CIN >> STR2;

   IF(!BRACKETS(STR1) || !BRACKETS(STR2)) {
      CERR << "[ERROR] ONE OF STRINGS ISN'T BRACKETS. ABORTING!" << ENDL;
      EXIT(1);
   }

   BRACKETREWORK(STR1);
   BRACKETREWORK(STR2);

   COUT << "BRACKET STRUCTURE [STR1]:  " << STR1 << ENDL;
   COUT << "BRACKET STRUCTURE [STR2]:  " << STR2 << ENDL;

   IF(STR1 != STR2)
      CERR << "STRINGS !EQUAL" << ENDL;
```

```
    ELSE
        COUT << "STRING 1 && 2 ARE STRUCTURE IDENTICAL" << ENDL;

    RETURN 0;
}

BOOL BRACKETS(STRING &STR) {
        STACK<CHAR> MSTACK;
        IF(STR[0] != '(')
            RETURN FALSE;

        FOR(UNSIGNED LONG IT=0; IT<=STR.SIZE(); IT++) {
            IF(IT == STR.SIZE()) {
                IF(!MSTACK.EMPTY())
                    RETURN FALSE;
                ELSE
                    RETURN TRUE;
            }

            IF(STR[IT] == '(')
                MSTACK.PUSH('(');
            ELSE IF(STR[IT] == ')' && !MSTACK.EMPTY()) {
                MSTACK.POP();
                IF(IT != STR.SIZE()-1 && MSTACK.EMPTY())
                    RETURN FALSE;
            }
        }
        RETURN TRUE;
}

VOID BRACKETREWORK(STRING &STR) {
    FOR(UNSIGNED LONG I=0; I<STR.SIZE();) {
        IF(STR[I] != '(' && STR[I] != ')') {
            STR.ERASE(STR.BEGIN() + STATIC_CAST<LONG>(I));
```

```
            CONTINUE;
        }
        I++;
    }
}
```