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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: Иерархические списки

Студент гр. 7383 _____ Александров Р.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2018

Цель работы.

Познакомиться с иерархическими списками и использованием их в практических задачах на языке программирования C++.

Постановка задачи.

Бинарное коромысло устроено так, что у него есть два плеча: левое и правое. Каждое плечо представляет собой (невесомый) стержень определенной длины, с которого свисает либо гирька, либо еще одно бинарное коромысло, устроенное таким же образом.

Вариант 1. Подсчитать общий вес заданного бинарного коромысла, то есть суммарный вес его гирек.

Реализация задачи.

Для решения поставленной задачи в работе были использованы 3 класса: Main, BinKor, Action.

В классе Main определяются функции для считывания данных:

- void consoleRead() - из консоли;
- void fileRead() - из файла.

Пользователю предлагается либо ввести данные вручную, либо указать текстовый файл, в котором они находятся.

В классе BinKor определяются иерархический список для хранения бинарного коромысла и функции взаимодействия с ним:

- int levelBk(std::stringstream &sstream) постепенно заполняет список;
- int getWeight() возвращает суммарный вес гирек бинарного коромысла.

В классе Action находятся 2 функции:

- void start(std::string str) принимает на вход строку, вызывает подсчет общего веса коромысла;

- std::string getResultString(std::string str) принимает исходную строку, создает промежуточную, через валидацию исходной заполняет промежуточную и возвращает ее в функцию void start(std::string str) .

Тестирование программы.

Программа собрана и проверена в операционных системах Xubuntu 18.04 с использованием компилятора g++ и Windows с использованием MinGW. В других ОС и компиляторах тестирование не проводилось. Тесты находятся в приложении А.

Вывод.

В ходе лабораторной работы были получены основные навыки программирования иерархических списков на языке C++, изучены приемы хранения бинарного коромысла в иерархическом списке. Результатом стала программа, рассчитывающая суммарный вес гирек в коромысле.

ПРИЛОЖЕНИЕ А
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Таблица 1 – Тестирование программы

Input	Output
((8((2 6)(4 3)))(1((5 5)(1 2))))	16
((2((4 8)(7 1)))(1 3))	12
((6((4 4)(8 8)))(9 6))	18
((-6((4 4)(8 8)))(9 -6))	Unexpected char [((-6((4 4)(8 8)))(9 -6))]
(()))3((2 4)(7 8)))(5 2))	Error. Wrong string format

ПРИЛОЖЕНИЕ Б
КОД ПРОГРАММЫ

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "main.h"
#include "binkor.h"

using namespace std;

void Main::fileRead() {
    string fileName, str;
    cout << "What`s the file name?" << endl;
    cin >> fileName;
    cout << "--------------" << endl;
    cout << "Reading from " << fileName << endl;
    cout << "--------------" << endl;

    ifstream inFile;
    inFile.open(fileName);
    if (!inFile) {
        cout << "Cannot find this file" << endl;
        cout << endl;
        return;
    }
    stringstream xstream;
    while (!inFile.eof()) {
        getline(inFile, str);
        xstream << str;
        BinKor::lisp s1;
        binKor.read_lisp(s1, xstream);
        int weight = binKor.getWeight(s1);
        cout << "weight = " << weight << endl;
        cout << "for list " << endl;
        binKor.write_lisp(s1);
        cout << endl;
        cout<<endl;
        binKor.destroy(s1);
    }
    inFile.close();
}

void Main::consoleRead() {
    string str;
    cout << "Enter string:" << endl;
    getline(cin, str); // remove '\n'
    getline(cin, str);
```

```
    stringstream xstream;
    xstream << str;
    BinKor::lisp s1;
    binKor.read_lisp(s1, xstream);
    int weight = binKor.getWeight(s1);
    cout << "weight = " << weight << endl;
    cout << "for list " << endl;
    binKor.write_lisp(s1);
    cout << endl;
    cout<<endl;
    binKor.destroy(s1);
}

void Main::menu() {
    cout << "1. Enter numbers from the txt" << endl;
    cout << "2. Enter numbers from the console" << endl;
    cout << "0. Exit" << endl;
}

int main() {
    cout << "Hello! This program calculates the weight of the binary
rocket" << endl;
    Main main;

    while (true) {
        main.menu();
        cin >> main.choice;
        switch (main.choice) {
            case 1:
                main.fileRead();
                break;
            case 2:
                main.consoleRead();
                break;
            case 0:
                exit(1);
        }
    }
}
```

main.h

```
#pragma once

#include "binkor.h"

class Main {
private:
    BinKor binKor;
public:
    Main() {}
```

```
    unsigned int choice;

    void consoleRead();

    void fileRead();

    void menu();
};
```

binkor.cpp

```
#include <iostream>
#include "binkor.h"


BinKor::s_expr *BinKor::head(const lisp s) {
    if (s != NULL) {
        return s->node.pair.hd;
    } else {
        return NULL;
    }
}

BinKor::s_expr *BinKor::tail(const lisp s) {
    if (s != NULL) {
        return s->node.pair.tl;
    } else {
        return NULL;
    }
}

bool BinKor::isAtom(const lisp s) {
    if (s == NULL)
        return false;
    else
        return (s->tag);
}

bool BinKor::isNull(const lisp s) {
    return s == NULL;
}

BinKor::lisp BinKor::cons(const lisp h, const lisp t) {
    lisp p;
    p = new s_expr;
    if (p == NULL) {
```

```
        cout << "Memory not enough" << endl;
        return nullptr;
    }
    else {
        p->tag = false;
        p->node.pair.hd = h;
        p->node.pair.tl = t;
        return p;
    }
}

BinKor::lisp BinKor::make_atom(const base x) {
    lisp s;
    s = new s_expr;
    s->tag = true;
    s->node.atom = x;
    return s;
}

void BinKor::read_lisp(lisp &y, stringstream &xstream) {
    base x;
    do
        xstream >> x;
    while (x == ' ');
    if (x)
        read_s_expr(x, y, xstream);
}

void BinKor::read_s_expr(base prev, lisp &y, stringstream &xstream) {
    if (prev == ')')  {
        cout << "Error: the initial brace is closing" << endl;
        return;
    }
    else if (prev != '(') {
        if (!isdigit(prev)) {
            cout << "Element " << prev << " it isn`t a digit, try again";
            exit(1);
        }
        y = make_atom(prev);
    } else read_seq(y, xstream);
}

void BinKor::read_seq(lisp &y, stringstream &xstream) {
    base x;
```

```
    lisp p1, p2;

    if (!(xstream >> x)) {
        cout << "Error: there is no closing bracket" << endl;
        return;
    }
    else {
        while (x == ' ')
            xstream >> x;
        if (x == ')') {
            y = NULL;
        } else {
            read_s_expr(x, p1, xstream);
            read_seq(p2, xstream);
            y = cons(p1, p2);
        }
    }
}

void BinKor::write_lisp(const lisp x) {
    if (isNull(x))
        cout << "()";
    else if (isAtom(x))
        cout << x->node.atom;
    else {
        cout << "(";
        write_seq(x);
        cout << ")";
    }
}

void BinKor::write_seq(const lisp x) {
    if (!isNull(x)) {
        write_lisp(head(x));
        write_seq(tail(x));
    }
}

int BinKor::getWeight(const lisp x) {
    if (isAtom(x)) {
        resultString.push_back(x->node.atom);
    } else {
        resultString.push_back('(');
        getWeighHelper(x);
```

```
        resultString.push_back(')');
        if   (resultString.at(resultString.size()   -   1)   ==   ')'   &&
resultString.at(resultString.size() - 2) != ')') {
            count += (resultString.at(resultString.length() - 2)) - '0';
        }
    }
    return count;
}

void BinKor::getWeighHelper(const lisp x) {
    if (!isNull(x)) {
        getWeight(head(x));
        getWeighHelper(tail(x));
    }
}

void BinKor::destroy(lisp s) {
    resultString.clear();
    count = 0;
    if (s != NULL) {
        if (!isAtom(s)) {
            destroy(head(s));
            destroy(tail(s));
        }
        delete s;
    }
}
```

binkor.h

```
#pragma once

#include <sstream>

using namespace std;

class BinKor {
public:
    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 *lisp;

    lisp head(const lisp s);

    lisp tail(const lisp s);

    lisp cons(const lisp h, const lisp t);

    lisp make_atom(const base x);


    bool isAtom(const lisp s);

    bool isNull(const lisp s);

    void destroy(lisp s);

    void read_lisp(lisp &y, stringstream &xstream);

    void read_s_expr(base prev, lisp &y, stringstream &xstream);

    void read_seq(lisp &y, stringstream &xstream);

    void write_lisp(const lisp x);

    void write_seq(const lisp x);


    int getWeight(const lisp x);

    void getWeighHelper(const lisp x);

private:
    int count = 0;
    string resultString;
};
```

Makefile

```
CXX=g++
RM=rm -f
LDFLAGS=-g -Wall

SRCS=main.cpp action.cpp binkor.cpp
OBJS=$(subst .cpp,.o,$(SRCS))
```

```
all: main

main: $(OBJS)
	$(CXX) $(LDFLAGS) -o main $(OBJS)

main.o: main.cpp main.h

action.o: action.cpp action.h

binkor.o: binkor.cpp binkor.h

clean:
	$(RM) $(OBJS)

distclean: clean
	$(RM) main
```