**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: Бинарное коромысло.

Вариант 4.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 5383 |  | Допира В. Е. |
| Преподаватель |  | Фирсов М. А. |

Санкт-Петербург

2016

**Лабораторная работа 1**

**Рекурсия**

**Вариант 4**

**1. Цели**

1. Изучить рекурсивный алгоритм;
2. Изучить бинарное коромысло и иерархические списки
3. Получить навыки работы с рекурсивными функциями.

Говорят, что бинарное коромысло сбалансировано, если момент вращения, действующий на его левое плечо, равен моменту вращения, действующему на правое плечо (то есть длина левого стержня, умноженная на вес груза, свисающего с него, равна соответствующему произведению для правой стороны), и если все подкоромысла, свисающие с его плеч, также сбалансированы.

**Задача:** Написать рекурсивную функцию или процедуру Balanced, которая проверяет заданное коромысло на сбалансированность (выдает значение true, если коромысло сбалансировано, и false в противном случае).

**2.Спецификация программы**

**2.1 Входные данные**

**Структуры данных:** в программе данные заданы с помощью структур:

struct Arm {

unsigned int length;

bool weightType; //false - гирька, true - бинарное коромысло

union {

unsigned int weight;

BinKor\* binkor;

};

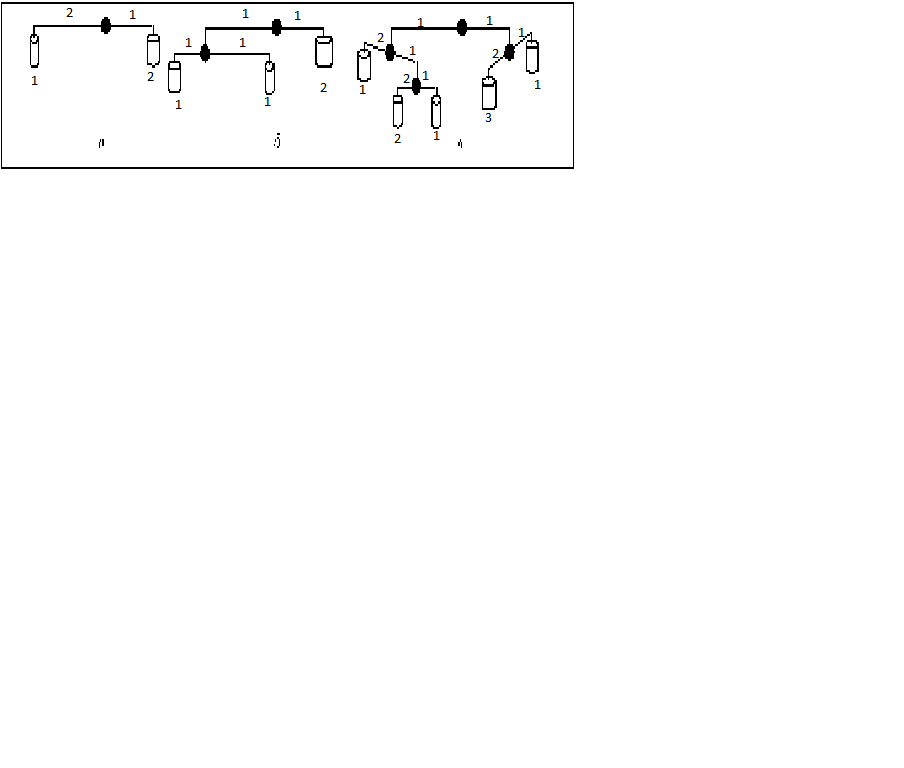
};

struct BinKor {

Arm left;

Arm right;

};

****

**2.2Выходные данные**

Программа выводит на экран результат проверки коромысла на сбалансированность, то есть сообщения: «Сбалансировано» или «Не сбалансировано».

**3.Алгоритм и структуры данных**

Программа выполняет следующую последовательность действий:

|  |
| --- |
| Функция чтения бинарного коромысла  {  Увеличение уровня;  Если (подробно){Вывод сообщения: «Чтение бинарного коромысла»}  Считывание элемента строки  Если (первый элемент не '(') {Вывод сообщения: «Ошибка обработки входных данных»  Выход из процедуры(1);  }  Если (подробно) { Вывод сообщения: «Чтение левого плеча»}  Чтение левого плеча  Если (подробно) { Вывод сообщения: «Чтение правого плеча»}  Чтение правого плеча  Считывание следующего элемента строки  Если (крайний элемент не ')') {Вывод сообщения: «Ошибка обработки входных данных»  Выход из процедуры(1);  }  Спуск по уровню  Возвращение элементов  }; |
| Функция вывода ошибок на входные данные  {  Чтение элементов  Если (первый элемент не '(') {Вывод сообщения: «Ошибка входных данных 3»  Выход из процедуры(1);  }  Чтение плеча  Если (Ошибка операций) { Вывод сообщения: «Ошибка обработки входных данных»  Выход из процедуры(1)  }  Если(длина плеча = 0){Вывод сообщения: «Ошибка входных данных :длина плеча равна нулю»  Выход из процедуры(1)  }  Если (подробно){ Вывод сообщения: «Длина плеча: »}  Чтение следующего элемента  Если (элемент не '(') {Вывод сообщения: «Груз: бинарное коромысло »  Чтение плеча с помощью функции readBK(in)  Выход из процедуры(1);  }  Иначе {  Чтение веса  Если (Ошибка операций) { Вывод сообщения: «Ошибка обработки входных данных 5»  Выход из процедуры(1)}  }  Чтение следующего элемента  Если (крайний элемент не ')') {Вывод сообщения: «Ошибка обработки входных данных 6»  Выход из процедуры(1);  }  Возвращение плеча  }; |
| Функция подсчета длины и веса  {  Сдвиг уровня вправо  Начальная длина = 0  Если (Левое плечо) {  Если (подробно) { Вывод сообщения: «Груз на левом плече - бинарное коромысло, рекурсивный вызов}  Работа с новым бинарным коромыслом  Вывод сообщения: «длина левого плеча=» }  Иначе{  Если (подробно) { Вывод сообщения: «Левое плечо: длина\*вес» }  Умножение длины на вес для левого плеча}  Если (Правое плечо) {  Если (подробно) { Вывод сообщения: «Груз на правом плече - бинарное коромысло, рекурсивный вызов}  Работа с новым бинарным коромыслом  Вывод сообщения: «длина левого плеча=» }  Иначе{  Если (подробно) { Вывод сообщения: «Правое плечо: длина\*вес» }  Умножение длины на вес для правого плеча}  }  Сдвиг уровня влево  Возвращение length;  } |
| Функция подсчета момента силы  {  Сдвиг уровня вправо  Начальная длина = 0  Если (Вес гирьки) {  Если (подробно) { Вывод сообщения: «Груз на левом плече - бинарное коромысло, рекурсивный вызов}  Умножение плеча на момент коромысла  Вывод сообщения: «длина левого плеча=» }  Иначе{  Если (подробно) { Вывод сообщения: «Левое плечо: длина\*вес» }  Умножение длины на вес для левого плеча}  Сдвиг уровня влево  Возвращение length;  } |
| Функция проверки на сбалансированность  { Если (Момент левого плеча = моменту правого) вернуть true;  Иначе вернуть false;  } |
| {  Подключение русского языка  Указатель пустой  Для (счетчика < числа аргументов)  {  Если (Элементы строки аргументов не 0) {  Подробно описать  } Иначе если (Элементы строки аргументов 0) {  Если (Счетчик >= числа аргументов) {  Вывод сообщения: «Ошибка в аргументах командной строки»  Выход из процедуры(1);  } Иначе {  Чтение из файла и вывод об этом сообщения  Если (файл не открыт) {  Вывод сообщения: «Ошибка в чтения файла»  Выход из процедуры(1);  }  }  }  }  Если (строка не нулевая) {  Если (подробно) { Вывод сообщения: «Чтение из консоли»}  Чтение элементов из консоли  Вывод сообщения: «Введите бинарное коромысло:»  }  Чтение бинарного коромысла с помощью функции readBK  Если (подробно) { Вывод сообщения: «Подсчет длины всех плеч:»}  Длины плеч с помощью функции Length  Если (коромысло сбалансировано после проверки функцией Balanced) Вывод сообщения: «Сбалансировано»  Иначе вывод сообщения: «НЕ сбалансировано»  } |

**Функции, используемые в программе**

BinKor\* readBK(istream &in)

Назначение: чтения бинарного коромысла

Параметры: in – строка входных данных

Возвращаемое значение: bk\_p – грузик бинарного коромысла

Arm readArm(istream &in)

Назначение: ошибки на входные данные

Параметры: in – строка входных данных

Возвращаемое значение: arm – плечо

unsigned int Length(const BinKor & bk)

Назначение: подсчет длины и веса

Параметры: bk – бинарное коромысло

Возвращаемое значение: arm – длина плеча

unsigned int ArmLength(const Arm & bkArm)

Назначение: подсчет момента силы

Параметры: bkArm – бинарное коромысло

Возвращаемое значение: arm – длина плеча

bool Balanced(const BinKor & bk)

Назначение: проверка на сбалансированность

Параметры: bk – бинарное коромысло

Возвращаемое значение: true или false- в зависимости от результата проверки

**4. Набор тестов**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Результат** | |
| **Бинарное коромысло** | **Ожидаемый** | **Полученный** |
| ((1 2)(1 2)) | Сбалансировано | Сбалансировано |
| ((15 30)(2 5)) | Не сбалансировано | Не сбалансировано |
| ((1 ((1 1) (1 1))) (1 1)) | Не сбалансировано | Не сбалансировано |
| ((1 ((1 1) (1 1))) (1 2)) | Сбалансировано | Сбалансировано |
| ((1 ((2 1) (1((2 2) (1 1))))) (1 ((1 1) (2 3)))) | Сбалансировано | Сбалансировано |
| ((1 ((20 1) (15((3 4) (1 1))))) (1 ((5 6) (2 3)))) | Не сбалансировано | Не сбалансировано |
| () | Ошибка обработки входных данных | Ошибка обработки входных данных 3 |
| ((15 30)(2 5) | Ошибка обработки входных данных | Ошибка обработки входных данных 2 |

1. **Текст программы:**

#include <cstdio>

#include <cstdlib> // exit

#include <cstring> // strcmp

#include <iostream>

#include <fstream>

using namespace std;

struct BinKor;

struct Arm;

Arm readArm(istream&);

BinKor\* readBK(istream&);

unsigned int bkWeight(const BinKor &);

int level = 0;

bool verbose = false;

//структуры данных

struct Arm {

unsigned int length;

bool weightType; //false - гирька, true - бинарное коромысло

union {

unsigned int weight;

BinKor\* binkor;

};

};

struct BinKor {

Arm left;

Arm right;

};

BinKor\* readBK(istream &in) //функция чтения бинарного коромысла

{

++level;

if (verbose){printf("(%d)%\*sЧтение бинарного коромысла\n", level, level,"");}

BinKor\* bk\_p = new BinKor;

in >> ws;

if (in.peek() != '(') {

printf("Ошибка обработки входных данных 1\n");

exit(1);

}

in.ignore();

if (verbose) { printf("(%d)%\*sЧтение левого плеча\n", level, level, ""); }

bk\_p->left = readArm(in);

if (verbose) { printf("(%d)%\*sЧтение правого плеча\n", level, level, ""); }

bk\_p->right = readArm(in);

in >> ws;

if (in.peek() != ')') {

printf("Ошибка обработки входных данных 2\n");

exit(1);

}

in.ignore();

--level;

return bk\_p;

};

Arm readArm(istream &in) //ошибки на входные данные

{

in >> ws;

Arm arm;

if (in.peek() != '(') {

printf("Ошибка обработки входных данных 3\n");

exit(1);

}

in.ignore();

in >> arm.length;

if (in.fail()) {

printf("Ошибка обработки входных данных 4\n");

exit(1);

}

if (arm.length == 0) {

printf("Ошибка во входных данных: длина плеча равна нулю\n");

exit(1);

}

if (verbose){printf("(%d)%\*sДлина плеча: %u\n", level, level,"",arm.length);}

in >> ws;

if (in.peek() == '(') {

arm.weightType = true;

if(verbose){printf("(%d)%\*sГруз: бинарное коромысло\n", level,level,"");}

arm.binkor = readBK(in);

} else {

arm.weightType = false;

in >> arm.weight;

if (in.fail()) {

printf("Ошибка обработки входных данных 5\n");

exit(1);

}

}

in >> ws;

if (in.peek() != ')') {

printf("Ошибка обработки входных данных 6\n");

exit(1);

}

in.ignore();

return arm;

};

unsigned int Length(const BinKor & bk) //подсчет длины и веса

{

++level;

short length = 0;

if (bk.left.weightType) {

if (verbose) { printf("(%d)%\*sГруз на левом плече - бинарное коромысло, рекурсивный вызов\n", level, level, ""); }

length += Length(\*bk.left.binkor) \* bk.left.length;

printf("(%d)%\*sдлина левого плеча=%d\n", level, level, "", length);

}

else{

if (verbose) { printf("(%d)%\*sЛевое плечо: длина\*вес - %hd\n", level, level, "", bk.left.length\* bk.left.weight ); }

length += bk.left.length\*bk.left.weight;

}

if (bk.right.weightType) {

if (verbose) { printf("(%d)%\*sГруз на правом плече - бинарное коромысло, рекурсивный вызов\n", level, level, ""); }

length += Length(\*bk.right.binkor)\*bk.right.length;

printf("(%d)%\*sдлина правого плеча=%d\n", level, level, "", length);

}

else{

if (verbose) { printf("(%d)%\*sПравое плечо: длина\*вес - %hd\n", level, level, "", bk.right.length\* bk.right.weight); }

length += bk.right.length\*bk.right.weight;

}

--level;

return length;

}

unsigned int ArmLength(const Arm & bkArm) //подсчет момента силы веса

{

++level;

short length = 0;

if (bkArm.weightType) {

if (verbose) { printf("(%d)%\*sГруз на левом плече - бинарное коромысло, рекурсивный вызов\n", level, level, ""); }

length += Length(\*bkArm.binkor) \* bkArm.length;

printf("(%d)%\*sдлина левого плеча=%d\n", level, level, "", length);

}

else{

if (verbose) { printf("(%d)%\*sЛевое плечо: длина\*вес - %hd\n", level, level, "", bkArm.length\* bkArm.weight ); }

length += bkArm.length\*bkArm.weight;

}

--level;

return length;

}

bool Balanced(const BinKor & bk) //проверка на сбалансированность

{ if (ArmLength(bk.left)==ArmLength(bk.right) )

return true;

else

return false;

}

int main(int argc, char const \*argv[])

{

setlocale(LC\_ALL,"Russian");

istream\* in = NULL;

for (int i = 1; i < argc; ++i) {

if (strcmp(argv[i], "-v") == 0) {

verbose = true;

} else if (strcmp(argv[i], "-f") == 0) {

if ((i + 1) >= argc) {

printf("Ошибка в аргументах командной строки\n");

exit(1);

} else {

ifstream\* fin = new ifstream(argv[i + 1]);

if (verbose) { printf("Чтение из файла %s:\n", argv[i + 1]); }

if (!fin->is\_open()) {

printf("Ошибка чтения файла\n");

exit(1);

}

in = fin;

}

}

}

if (in == NULL) {

if (verbose) { printf("Чтение из консоли\n"); }

in = &cin;

printf("Введите бинарное коромысло: ");

}

BinKor\* bk;

bk = readBK(\*in);

if (verbose) { printf("\nПодсчет длины всех плеч:\n"); }

int bkw = Length( \* bk);

if (Balanced(\* bk))

printf("Сбалансировано\n");

else

printf("НЕ сбалансировано\n");

}

**6. Вывод**

В ходе данной работы были получены практические навыки работы с иерархическом списком на языке программирования С++.