**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

Курсовая РАБОТА

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1309 |  | Фомин В. В. |
| Преподаватель |  | Калмычков В.А |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Фомин В. В. | | |
| Группа 1309 | | |
| Исходные данные: Входной файл с названием делянок и массой дров на них.  Необходимо: Написать программу, моделирующую поездку грузовика, перевозящего дрова с делянок в кочегарку, с максимальным использованием грузоподъемности грузовика в каждый момент. | | |
| Содержание пояснительной записки: «Постановка задачи», «Анализ задачи», «Описание метода», «Описание алгоритма», «Структура данных», «Оценка быстродействия (эффективности)», «Реализация задачи», «Блок-схемы алгоритмов», «Текст программы» «Тестирование», «Заключение» | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Фомин В. В. |
| Преподаватель |  | Калмычков В.А |

**Аннотация**

В работе представлена программа, решающая поставленную задачу по перевозке дров с подготовленным функционалом. На вход подается файл, с названием делянки и массой дров на ней. Программой ведётся моделирование перевозки, выбор дров для транспортировки в каждый момент.

**Summary**

The paper presents a program that solves the task of transporting firewood with prepared functionality. A file is submitted to the input, with the name of the sawmill and the mass of firewood on it. The program simulates transportation, the choice of firewood for transportation at every moment.

Оглавление

[Цель курсовой работы: 5](#_Toc105363336)

[Исходная постановка: 5](#_Toc105363337)

[Способы реализации ввода-вывода 5](#_Toc105363338)

[Написанные функции: 5](#_Toc105363339)

[Разбиение программы на файлы 10](#_Toc105363340)

[Алгоритмы и структуры данных 11](#_Toc105363341)

[Текст программы 15](#_Toc105363342)

[Тестирование программы 23](#_Toc105363343)

[Вывод 23](#_Toc105363344)

# **Цель курсовой работы:**

Продемонстрировать умение хранить и обрабатывать списки и умение решать задачи методом динамического программирования.

# Исходная постановка:

Изображение выглядит как текст

Автоматически созданное описание

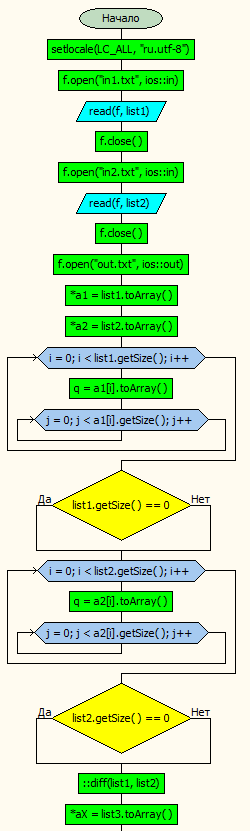
Способы реализации ввода-вывода:

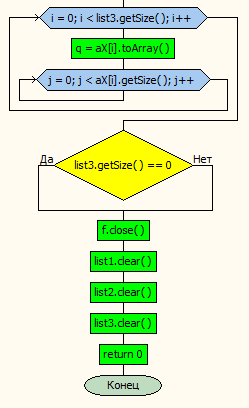
|  |  |  |
| --- | --- | --- |
|  | iostream | fstream |
| Ввод | cin | ios::in |
| Вывод | cout | ios::out |

# **Написанные функции:**

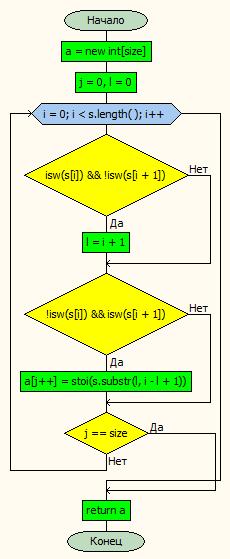
|  |  |
| --- | --- |
| isw | Проверка на пробельный символ |
| sizeOfElements | Число элементов в строке |
| split | Разбить строку на несколько строк по пробелам |
| toString | Сделать строку из списка блоков |
| getSize | Длина списка |
| add | Добавить элемент |
| get | Вернуть элемент по индексу |
| set | Заменить элемент по индексу |
| remove | Удалить элемент по индексу |
| copy | Скопировать список |
| clear | Очистить список |
| toArray | Вернуть массив из элементов списка |
| calc | Создать матрицу комбинаций забираемых с делянок бревен |
| traceResult | Выбрать бревна опираясь на матрицу |

**main:**

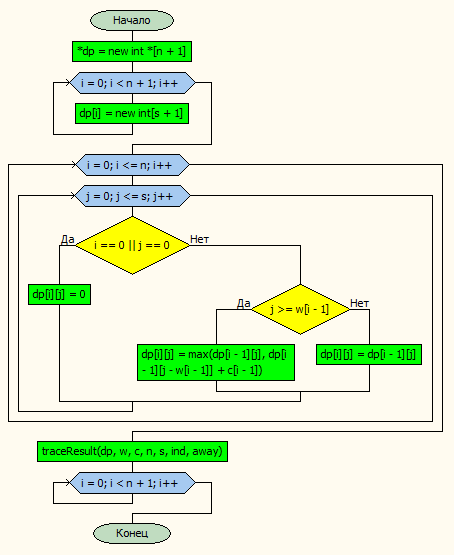
****

****

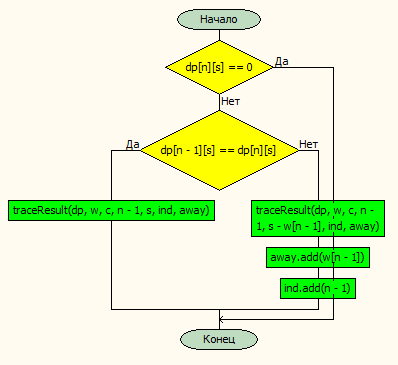
**split:**

****

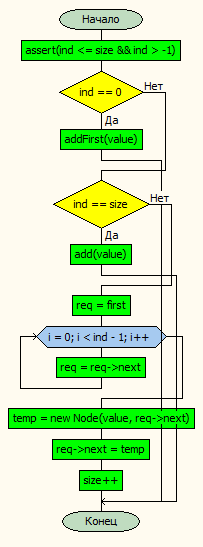
**calc:**

****

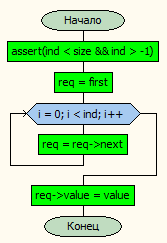
**traceResult:**

****

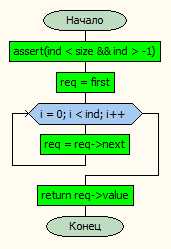
**add:**

****

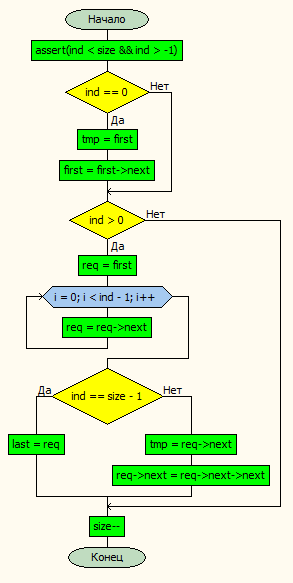
**get:**

****

**set:**

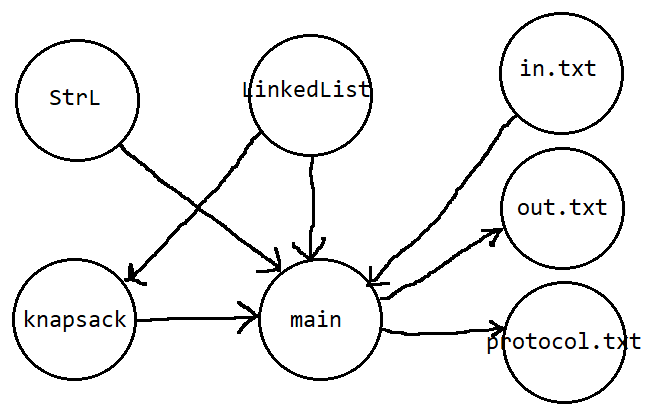
****

**remove:**

****

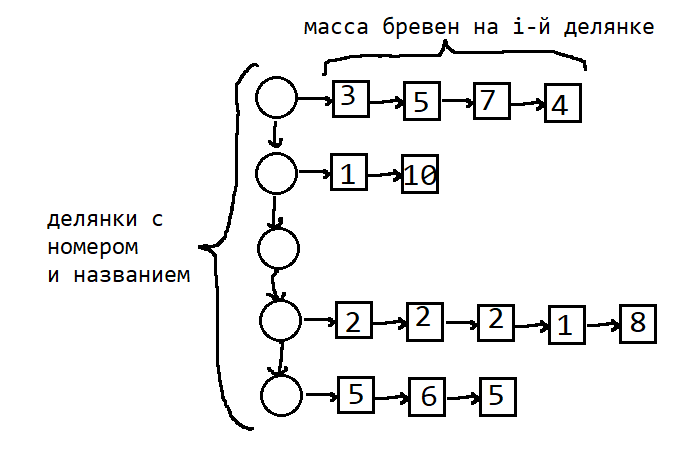
# **Разбиение программы на файлы**

|  |  |
| --- | --- |
| main.cpp | Основа программы |
| StrL.cpp | Структура слова, из которых состоит список |
| StrL.h | Интерфейс структуры слова |
| LinkedList.h | Интерфейс списка вместе с реализацией |
| LinkedList.cpp | Остается пустым, так как шаблонные классы реализуются в заголовочном файле |
| Knapsack.cpp | Реализация алгоритма «рюкзака» |
| Knapsack.h | Заголовок алгоритма «рюкзака» |

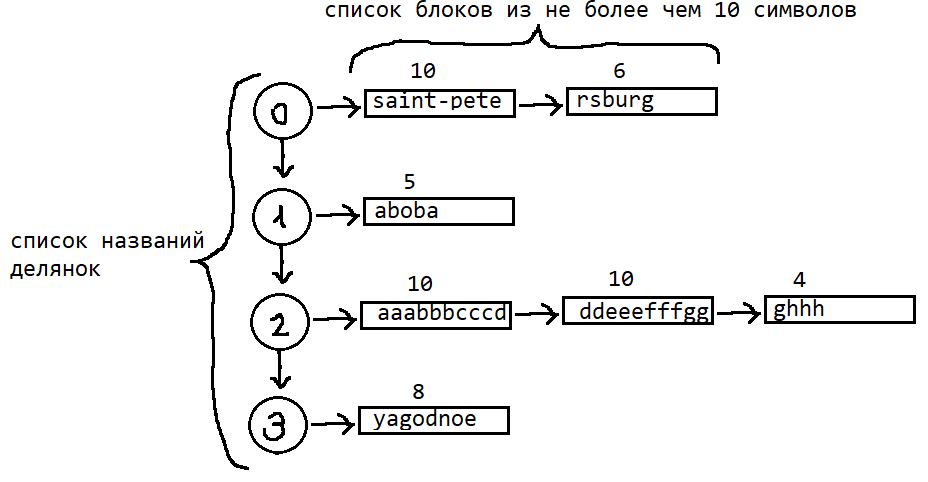


# Алгоритмы и структуры данных

**Список list:**



**Список names:**



Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

# Текст программы

**Main**

#include <iostream>  
#include <fstream>  
#include "LinkedList.h"  
#include "Knapsack.h"  
#include <string>  
#include "StrL.h"  
  
bool isw(int x) {  
 return x == '\0' || x == ' ' || x == '\n';  
}  
  
int sizeOfElements(string s) {  
 int size = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (!isw(s[i]) && isw(s[i + 1])) {  
 size++;  
 }  
 }  
 return size;  
}  
  
int \*split(string s, int size) {  
 int \*a = new int[size];  
 int j = 0, l = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (isw(s[i]) && !isw(s[i + 1])) {  
 l = i + 1;  
 }  
 if (!isw(s[i]) && isw(s[i + 1])) {  
 a[j++] = stoi(s.substr(l, i - l + 1));  
 }  
 if (j == size) {  
 break;  
 }  
 }  
 return a;  
}  
  
string toString(LinkedList<StrL> list){  
 string s = "";  
 for (int i = 0; i < list.getSize(); ++i) {  
 s += list.get(i).arr;  
 }  
 return s;  
}  
  
int main() {  
 const int CAPACITY = 10, N = 10;  
 LinkedList<LinkedList<int>> list;  
 LinkedList<LinkedList<StrL>> names;  
 LinkedList<int> stoke;//кочегарка  
 LinkedList<int> track;//грузовик  
 LinkedList<int> stay;//осталось на делянке  
 LinkedList<int> ind;//индексы дров которые забрали из stay  
  
 fstream fin;  
 fin.open("in.txt", ios::in);  
 fstream fout;  
 fout.open("out.txt", ios::out);  
 fstream ptc;  
 ptc.open("protocol.txt", ios::out);  
  
 int n = 0;  
 string s;  
 while (!fin.eof()) {  
 //название  
 getline(fin, s);  
 LinkedList<StrL> name;  
 for (int i = 0; i < (s.length() + N - 1) / N; i++) {  
 name.add(StrL(s.substr(i \* N, N)));  
 }  
 names.add(name);  
 //дрова  
 getline(fin, s);  
 int k = sizeOfElements(s);  
 int \*a = split(s, k);  
 LinkedList<int> tmp;  
 for (int i = 0; i < k; i++) {  
 tmp.add(a[i]);  
 n++;  
 }  
 delete[] a;  
 list.add(tmp);  
 }  
 fin.close();  
  
 int way = 1;  
 while (n > 0) {  
 fout << "объезд №" << way << ") ";  
 ptc << "объезд №" << way << ") ";  
 way++;  
 for (int i = 0; i < list.getSize(); i++) {  
 stay = list.get(i);  
 for (int j = 0; j < track.getSize(); ++j) {  
 stay.add(track.get(j));  
 }  
 track.clear();  
 int \*w = stay.toArray();  
 calc(w, w, stay.getSize(), CAPACITY, ind, track);  
 for (int j = 0; j < ind.getSize(); ++j) {  
 stay.remove(ind.get(j) - j);  
 }  
 ind.clear();  
 list.set(stay, i);  
 fout << "делянка " << toString(names.get(i)) << ": ";  
 for (int j = 0; j < stay.getSize(); j++) {  
 fout << stay.get(j) << ", ";  
 }  
 fout << "; ";  
 ptc << "грузовик увез с делянки " << toString(names.get(i)) << ": ";  
 for (int j = 0; j < track.getSize(); j++) {  
 ptc << track.get(j) << ", ";  
 }  
 ptc << "; ";  
 delete[] w;  
 }  
 stoke.insertList(track, stoke.getSize());  
 fout << "кочегарка: ";  
 for (int j = 0; j < stoke.getSize(); j++) {  
 fout << stoke.get(j) << ", ";  
 }  
 fout << "\n";  
 ptc << "грузовик привез в кочегарку: ";  
 for (int j = 0; j < track.getSize(); j++) {  
 ptc << track.get(j) << ", ";  
 }  
 ptc << "\n";  
 n -= track.getSize();  
 track.clear();  
 }  
 fout.close();  
 list.clear();  
 track.clear();  
 stoke.clear();  
 return 0;  
}

**LinkedList.h**

#pragma once  
  
#include <iostream>  
#include <cassert>  
  
using namespace std;  
  
template<typename T>  
class LinkedList {  
private:  
 struct Node {  
 T value;  
 Node \*next;  
  
 Node(T value, Node \*next) {  
 this->value = value;  
 this->next = next;  
 };  
 };  
  
 int size = 0;  
 Node \*first;  
 Node \*last;  
public:  
 //~LinkedList();  
  
 int getSize();  
  
 void addFirst(T value);  
  
 void add(T value);  
  
 void add(T value, int ind);  
  
 void remove(int ind);  
  
 void set(T value, int ind);  
  
 T get(int ind);  
  
 LinkedList<T> copy();  
  
 void clear();  
  
 void print();  
  
 T \*toArray();  
  
 void insertList(LinkedList<T> a, int ind);  
};  
  
template<typename T>  
int LinkedList<T>::getSize() {  
 return size;  
}  
  
template<typename T>  
void LinkedList<T>::addFirst(T value) {  
 if (size > 0) {  
 first = new Node(value, first);  
 } else {  
 first = new Node(value, nullptr);  
 last = first;  
 }  
 size++;  
}  
  
template<typename T>  
void LinkedList<T>::add(T value) {  
 if (size > 0) {  
 last->next = new Node(value, nullptr);  
 last = last->next;  
 } else {  
 first = new Node(value, nullptr);  
 last = first;  
 }  
 size++;  
}  
  
template<typename T>  
void LinkedList<T>::add(T value, int ind) {  
 assert(ind <= size && ind > -1);  
 if (ind == 0) {  
 addFirst(value);  
 return;  
 }  
 if (ind == size) {  
 add(value);  
 return;  
 }  
 Node \*req = first;  
 for (int i = 0; i < ind - 1; i++) {  
 req = req->next;  
 }  
 Node \*temp = new Node(value, req->next);  
 req->next = temp;  
 size++;  
  
}  
  
template<typename T>  
void LinkedList<T>::set(T value, int ind) {  
 assert(ind < size && ind > -1);  
 Node \*req = first;  
 for (int i = 0; i < ind; i++) {  
 req = req->next;  
 }  
 req->value = value;  
}  
  
template<typename T>  
T LinkedList<T>::get(int ind) {  
 assert(ind < size && ind > -1);  
 Node \*req = first;  
 for (int i = 0; i < ind; i++) {  
 req = req->next;  
 }  
 return req->value;  
}  
  
template<typename T>  
LinkedList<T> LinkedList<T>::copy() {  
 LinkedList<T> b;  
 Node \*req = first;  
 for (int i = 0; i < size; i++) {  
 b.add(req->value);  
 req = req->next;  
 }  
 return b;  
}  
  
template<typename T>  
void LinkedList<T>::clear() {  
 while (size > 0) {  
 Node \*second = first->next;  
 delete first;  
 first = second;  
 size--;  
 }  
}  
  
template<typename T>  
void LinkedList<T>::print() {  
 Node \*req = first;  
 for (int i = 0; i < size; i++) {  
 cout << (req->value);  
 req = req->next;  
 }  
}  
  
template<typename T>  
T \*LinkedList<T>::toArray() {  
 T \*a = new T[getSize()];  
 Node \*req = first;  
 for (int i = 0; i < size; i++) {  
 a[i] = (req->value);  
 req = req->next;  
 }  
 return a;  
}  
  
/\*  
template<typename T>  
LinkedList<T>::~LinkedList() {  
 clear();  
}  
\*/  
template<typename T>  
void LinkedList<T>::remove(int ind) {  
 //cout << ind << "\n";  
 assert(ind < size && ind > -1);  
 if (ind == 0) {  
 Node \*tmp = first;  
 first = first->next;  
 delete tmp;  
 }  
 if (ind > 0) {  
 Node \*req = first;  
 for (int i = 0; i < ind - 1; i++) {  
 req = req->next;  
 }  
 if (ind == size - 1) {  
 last = req;  
 delete req->next;  
 } else {  
 Node \*tmp = req->next;  
 req->next = req->next->next;  
 delete tmp;  
 }  
 }  
 size--;  
}  
  
template<typename T>  
void LinkedList<T>::insertList(LinkedList<T> a, int ind) {  
 assert(ind <= size && ind > -1);  
 LinkedList<T> b = a.copy();  
 if (size == 0) {  
 first = b.first;  
 last = b.last;  
 size = b.size;  
 } else {  
 if (ind == 0) {  
 first = b.first;  
 b.last->next = first;  
 } else if (ind == size) {  
 last->next = b.first;  
 last = b.last;  
 } else {  
 Node \*req = first;  
 for (int i = 0; i < ind - 1; i++) {  
 req = req->next;  
 }  
 b.last->next = req->next;  
 req->next = b.first;  
 }  
 size += b.size;  
 }  
}

**StrL.h**

#pragma once  
  
#include <string>  
#include <iostream>  
  
using namespace std;  
  
struct StrL {  
 char \*arr;  
 int len;  
  
 StrL(string s);  
  
 StrL();  
  
 int compareStrl(StrL s);  
};

**StrL.cpp**

#include "StrL.h"  
  
using namespace std;  
  
StrL::StrL(string s) {  
 arr = new char[s.length() + 1];  
 strcpy(arr, s.c\_str());  
 len = s.length();  
}  
  
StrL::StrL() {  
  
}  
  
int StrL::compareStrl(StrL s) {  
 for (int i = 0; i < min(len, s.len); i++) {  
 if (arr[i] < s.arr[i]) {  
 return -1;  
 }  
 if (arr[i] > s.arr[i]) {  
 return 1;  
 }  
 }  
 if (len < s.len) {  
 return -1;  
 }  
 if (len > s.len) {  
 return 1;  
 }  
 return 0;  
}

**Knapsack.h**

#pragma once  
#include <iostream>  
#include "LinkedList.h"  
#include <fstream>  
  
void traceResult(int \*\*dp, int \*w, int \*c, int n, int s, LinkedList<int> &ind, LinkedList<int> &away);  
  
void calc(int \*w, int \*c, int n, int s, LinkedList<int> &ind, LinkedList<int> &away);

**Knapsack.cpp**

#include "Knapsack.h"  
  
using namespace std;  
  
void traceResult(int \*\*dp, int \*w, int \*c, int n, int s, LinkedList<int> &ind, LinkedList<int> &away) {  
 if (dp[n][s] == 0) {  
 return;  
 }  
 if (dp[n - 1][s] == dp[n][s]) {  
 traceResult(dp, w, c, n - 1, s, ind, away);  
 } else {  
 traceResult(dp, w, c, n - 1, s - w[n-1], ind, away);  
 away.add(w[n-1]);  
 ind.add(n-1);  
 }  
}  
  
void calc(int \*w, int\*c, int n, int s, LinkedList<int> &ind, LinkedList<int> &away) {  
 int \*\*dp = new int \*[n + 1];  
 for (int i = 0; i < n + 1; i++) {  
 dp[i] = new int[s + 1];  
 }  
  
 for (int i = 0; i <= n; i++) {  
 for (int j = 0; j <= s; j++) {  
 if (i == 0 || j == 0) {  
 dp[i][j] = 0;  
 } else {  
 if (j >= w[i-1]) {  
 dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[i-1]] + c[i-1]);  
 } else {  
 dp[i][j] = dp[i - 1][j];  
 }  
 }  
 }  
 }  
 traceResult(dp, w, c, n, s, ind, away);  
 for (int i = 0; i < n + 1; i++) {  
 delete[] dp[i];  
 }  
 delete[] dp;  
 //return dp[n][s];  
}

# Тестирование программы

in.txt

alpha  
1 2 3 4 5  
betta  
1 1 1  
gamma  
1 1  
phi  
1 1 1 1 1 10  
tetta  
1 1 1

out.txt

объезд №1) делянка alpha: 5, ; делянка betta: 3, ; делянка gamma: 2, ; делянка phi: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, ; делянка tetta: 1, 1, 1, ; кочегарка: 10,   
объезд №2) делянка alpha: ; делянка betta: ; делянка gamma: ; делянка phi: 1, 4, 2, 3, 5, ; делянка tetta: 1, 1, 1, ; кочегарка: 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,   
объезд №3) делянка alpha: ; делянка betta: ; делянка gamma: ; делянка phi: 5, ; делянка tetta: 3, ; кочегарка: 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 2,   
объезд №4) делянка alpha: ; делянка betta: ; делянка gamma: ; делянка phi: ; делянка tetta: ; кочегарка: 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 2, 3, 5,

protocol.txt

объезд №1) грузовик увез с делянки alpha: 1, 2, 3, 4, ; грузовик увез с делянки betta: 1, 1, 1, 1, 2, 4, ; грузовик увез с делянки gamma: 1, 1, 1, 1, 1, 1, 4, ; грузовик увез с делянки phi: 10, ; грузовик увез с делянки tetta: 10, ; грузовик привез в кочегарку: 10,   
объезд №2) грузовик увез с делянки alpha: 5, ; грузовик увез с делянки betta: 3, 5, ; грузовик увез с делянки gamma: 2, 3, 5, ; грузовик увез с делянки phi: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ; грузовик увез с делянки tetta: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ; грузовик привез в кочегарку: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,   
объезд №3) грузовик увез с делянки alpha: ; грузовик увез с делянки betta: ; грузовик увез с делянки gamma: ; грузовик увез с делянки phi: 1, 4, 2, 3, ; грузовик увез с делянки tetta: 1, 1, 1, 1, 4, 2, ; грузовик привез в кочегарку: 1, 1, 1, 1, 4, 2,   
объезд №4) грузовик увез с делянки alpha: ; грузовик увез с делянки betta: ; грузовик увез с делянки gamma: ; грузовик увез с делянки phi: 5, ; грузовик увез с делянки tetta: 3, 5, ; грузовик привез в кочегарку: 3, 5,

# Вывод

Цель работы выполнена, программа написана с достаточно удобным интерфейсом. В ходе написания программы были закреплены практические навыки программирования, обобщены знания синтаксиса языка, и продемонстрировано усвоение материала за второй семестр. Поставленная цель полностью соответствует полученному результату.