14Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский экономический университет имени Г.В. Плеханова»   
**Московский приборостроительный техникум**

**Отчет**   
о выполнении практической работы № 14

Дисциплина: ОП.04. Основы алгоритмизации и программирования

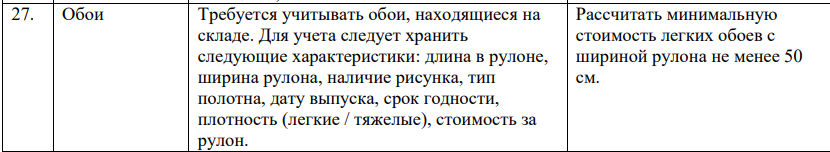
Тема: Использование указателей для организации связанных списков

Специальность: 09.02.07 Информационные системы и программирование

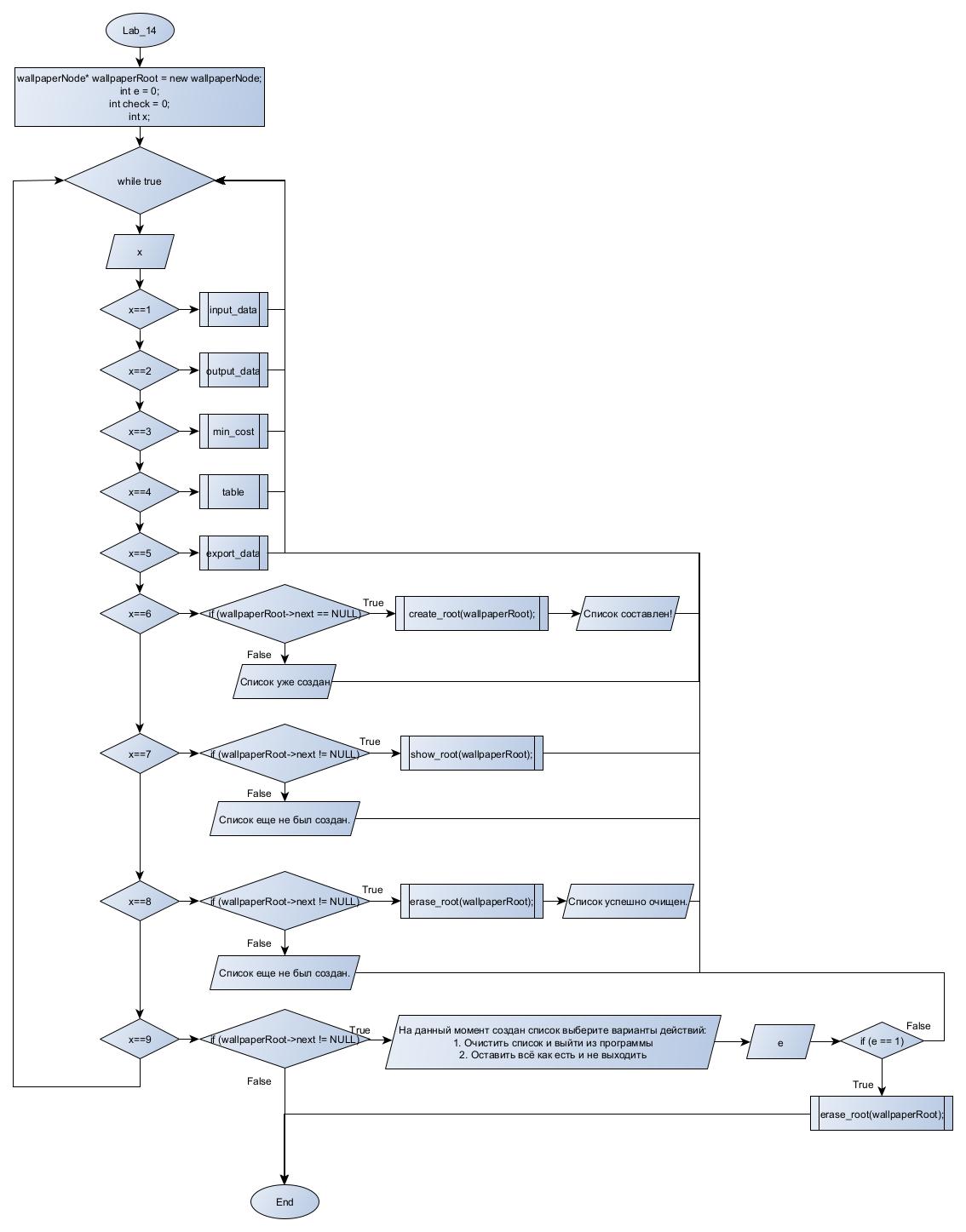
Квалификация: Специалист по информационным системам

Выполнил: Dusky Fox  
Студент группы:   
Вариант: № 27  
Дата: «26» января 2022 г.

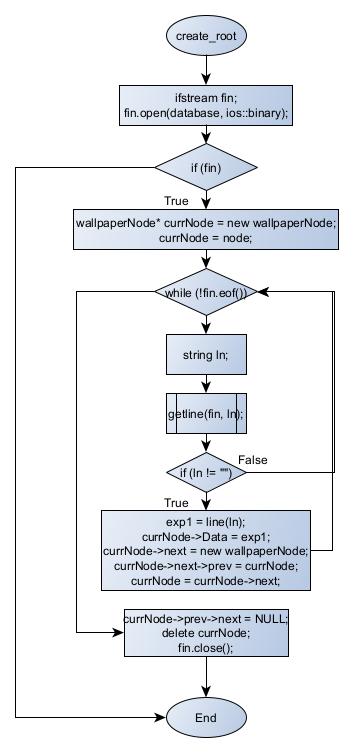
Проверил:   
Преподаватель: Клопов Д.А.  
Дата: «\_\_\_» \_\_\_\_\_\_\_\_\_ 202\_\_ г.   
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



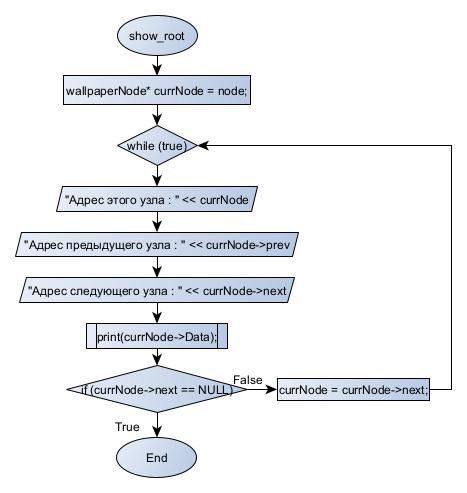
1 Задание



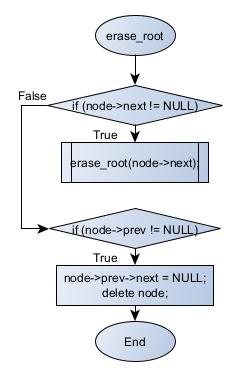
2 Тело программы



3 Функция создания динамического списка



4 Функция отображения динамического списка



5 Функция очистки динамического списка

# Файл Lab\_14.cpp

#include "wallpaper.h"

wallpaperNode\* wallpaperRoot = new wallpaperNode;

int main(string args[])

{

string a = "12,4";

setlocale(LC\_ALL, "Ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int e = 0;

int check = 0;

int x;

while (true) {

system("cls");

cout << "1. Запись данных в файл " << endl << "2. Вывод данных из файла" << endl << "3. Найти минимальную стоимость лёгкого рулона с шириной от 50 см" << endl << "4. Вывод в файл таблицы" << endl << "5. Вывод в формате .CSV" << endl << "---------------------------" << endl << "6. Составить список из файла" << endl << "7. Показать список" << endl << "8. Очистить список" << endl << "---------------------------" << endl << "9. Выход" << endl << "---------> ";

cin >> x;

switch (x) {

case 1:

input\_data();

break;

case 2:

output\_data();

system("pause");

break;

case 3:

min\_cost();

system("pause");

break;

case 4:

check = table();

if (check == 0) {

cout << "Данные были успешно перенесены в таблицу!\n";

cout << "Открыть таблицу? (1 - да/ 0 - нет)";

cin >> e;

if (e == 1) system("start wallpaper.txt");

}

else if (check == 1) {

cout << "Не удалось открыть файл данных";

}

else cout << "Не удалось создать таблицу";

break;

case 5:

check = export\_data();

if (check == 0) {

cout << "Данные были успешно перенесены в .CSV таблицу!\n";

cout << "Открыть таблицу? (1 - да/ 0 - нет)";

cin >> e;

if (e == 1) system("start excel wallpaper.csv");

}

else if (check == 1) {

cout << "Не удалось открыть файл данных";

}

else cout << "Не удалось создать .CSV таблицу";

break;

case 6:

if (wallpaperRoot->next == NULL) {

create\_root(wallpaperRoot);

cout << "Список составлен!" << endl;

}

else

cout << "Список уже создан" << endl;

system("pause");

break;

case 7:

if (wallpaperRoot->next != NULL)

show\_root(wallpaperRoot);

else

cout << "Список еще не был создан." << endl;

system("pause");

break;

case 8:

if (wallpaperRoot->next != NULL)

{

erase\_root(wallpaperRoot);

cout << "Список успешно очищен." << endl;

}

else

cout << "Список еще не был создан." << endl;

system("pause");

break;

case 9:

if (wallpaperRoot->next != NULL) {

cout << "!----------------------!\n" << "На данный момент создан список выберите варианты действий:\n" << " 1. Очистить список и выйти из программы\n" << " 2. Оставить всё как есть и не выходить\n" << "------>";

cin >> e;

if (e == 1) {

erase\_root(wallpaperRoot);

return 0;

}

else break;

}

return 0;

break;

default:

cout << endl << "Вы ввели неверное значение." << endl;

system("timeout 5");

}

}

}

# Файл wallpaper.cpp

#include "wallpaper.h"

wallpaper exp1;

// Вывод данных на экран

void print(wallpaper pr) {

cout << "Длина: " << pr.length << endl;

cout << "Ширина: " << pr.width << endl;

cout << "Есть ли рисунок: " << pr.picture << endl;

cout << "Тип полотна: " << pr.canvas << endl;

cout << "Дата изготовления: " << pr.date << endl;

cout << "Срок годности (в годах): " << pr.expiration\_date << endl;

cout << "Плотность: " << pr.density << endl;

cout << "Цена за рулон: " << pr.cost << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

// Получение данных из файла

wallpaper line(string sm) {

wallpaper ret;

const char p = ' ';

string put[8];

int trigger = 0;

for (int i = 0; i < sm.size(); i++) {

if (sm[i] != p) {

put[trigger] += sm[i];

}

else {

trigger++;

}

}

ret.length = stoi(put[0]);

ret.width = stoi(put[1]);

ret.picture = put[2];

ret.canvas = put[3];

ret.date = put[4];

ret.expiration\_date = stoi(put[5]);

ret.density = put[6];

ret.cost = stof(put[7]);

return ret;

}

// Поиск минимальной стоимости и вывод данных на экран

void min\_cost() {

float min = 32000000;

ifstream fin;

cout << endl;

fin.open(database, ios::in | ios::binary);

if (!fin) {

cout << "База данных пуста" << endl;

system("timeout 5");

return;

}

byte a = 0;

while (!fin.eof()) {

string ln = "";

getline(fin, ln);

if (ln != "") {

exp1 = line(ln);

print(exp1);

if (((exp1.density == "легкое" || exp1.density == "лёгкое") && (exp1.width >= 50) && min > exp1.cost))

min = exp1.cost;

}

}

cout << endl;

if (min != 32000000) {

cout << "Минимальная стоимость за рулон лёгких обоев с шириной рулона не менее 50 см составляет " << min << " руб.\n";

}

else {

cout << "Рулон лёгких обоев с шириной рулона не менее 50 см не был найден.\n";

}

return;

}

// Ввод данных

void input\_data() {

cin.ignore();

cout << "Введите длину рулона (см): ";

cin >> exp1.length;

cout << "Введите ширину рулона (см): ";

cin >> exp1.width;

cout << "Есть ли рисунок на обоях (да/нет): ";

cin >> exp1.picture;

cout << "Введите тип полотна: ";

cin >> exp1.canvas;

cout << "Введите дату изготовления (дд.мм.гггг) : ";

cin >> exp1.date;

cout << "Введите срок годности полотна: ";

cin >> exp1.expiration\_date;

cout << "Введите плотность полотна (лёгкое/тяжёлое): ";

cin >> exp1.density;

cout << "Введите стоимость за рулон (руб,коп): ";

cin >> exp1.cost;

ofstream fout;

fout.open(database, ios::app | ios::binary);

fout.seekp(0, ios::end);

fout << exp1.length << " " << exp1.width << " " << exp1.picture << " " << exp1.canvas << " " << exp1.date << " " << exp1.expiration\_date << " " << exp1.density << " " << exp1.cost << endl;

fout.close();

}

// Вывод данных на экран

void output\_data() {

ifstream fin;

cout << endl;

fin.open(database, ios::in | ios::binary);

if (!fin) {

cout << "База данных пуста" << endl;

return;

}

while (!fin.eof()) {

string ln;

getline(fin, ln);

if (ln != "") {

exp1 = line(ln);

print(exp1);

}

}

fin.close();

}

// Вывод данных в таблицу txt

int table() {

ifstream fin;

ofstream fout;

fout.open(txt\_table, ios::out);

fin.open(database, ios::binary);

if (!fout) {

return 2;

}

if (fin) {

string title = "| Длина | Ширина | Рисунок | Тип полотна | Дата изготовления | Срок годности | Плотность | Цена за рулон |";

string delim = "+------------ + ------------ + ------- + --------------------- + ----------------- + ------------- + --------------- + -----------------+";

fout.setf(ios::fixed);

fout.precision(2);

fout << delim << endl;

fout << title << endl;

fout << delim << endl;

while (!fin.eof()) {

const char p = ' ';

string sm;

string put[8];

int trigger = 0;

getline(fin, sm);

if (sm != "") {

for (int i = 0; i < sm.size(); i++) {

if (sm[i] != p) {

put[trigger] += sm[i];

}

else {

trigger++;

}

}

fout.setf(ios::right);

fout << "|"; fout.width(13); fout << put[0]; // Длина

fout << "|"; fout.width(14); fout << put[1]; // Ширина

// Рисунок

if (put[2] == "1") {

fout << "|"; fout.width(9); fout << "Да";

}

else {

fout << "|"; fout.width(9); fout << "Нет";

}

fout << "|"; fout.width(23); fout << put[3]; // Тип полотна

fout << "|"; fout.width(19); fout << put[4]; // Дата изготовления

fout << "|"; fout.width(15); fout << put[5]; // Срок годности

fout << "|"; fout.width(17); fout << put[6]; // Плотность

fout << "|"; fout.width(18); fout << put[7]; // Стоимость

fout << "|" << endl;

}

}

fout << delim << endl;

fout.close();

fin.close();

return 0;

}

else return 1;

}

// Экспорт в CSV

int export\_data() {

ifstream fin;

ofstream fout;

fout.open(csv\_table, ios::out);

fin.open(database, ios::binary);

string title = "Длина;Ширина;Рисунок;Тип полотна;Дата изготовления;Срок годности;Плотность;Цена за рулон";

fout << title << endl;

if (!fout) {

return 2;

}

if (fin) {

while (!fin.eof()) {

const char p = ' ';

string sm;

string put[8];

int trigger = 0;

getline(fin, sm);

if (sm != "") {

for (int i = 0; i < sm.size(); i++) {

if (sm[i] != p) {

put[trigger] += sm[i];

}

else {

trigger++;

}

}

fout << put[0] << ";"; //Длина

fout << put[1] << ";"; // Ширина

//Рисунок

if (put[2] == "1") {

fout << "Да" << ";";

}

else {

fout << "Нет" << ";";

}

fout << put[3] << ";"; // Тип полотна

fout << put[4] << ";"; // Дата изготовления

fout << put[5] << ";"; // Срок годности

fout << put[6] << ";"; // Плотность

string price = "";

for (int i = 0; i < put[7].size(); i++) {

if (put[7][i] != '.') {

price += put[7][i];

}

else {

price += ',';

}

}

fout << price << endl; // Цена за рулон

}

}

fout.close();

fin.close();

return 0;

}

else return 1;

}

// Создание динамического списка

void create\_root(wallpaperNode\* node) {

ifstream fin;

fin.open(database, ios::binary);

if (fin) {

wallpaperNode\* currNode = new wallpaperNode;

currNode = node;

while (!fin.eof()) {

string ln;

getline(fin, ln);

if (ln != "") {

exp1 = line(ln);

currNode->Data = exp1;

currNode->next = new wallpaperNode;

currNode->next->prev = currNode;

currNode = currNode->next;

}

}

currNode->prev->next = NULL;

delete currNode;

fin.close();

}

}

// Вывод списка

void show\_root(wallpaperNode\* node) {

wallpaperNode\* currNode = node;

while (true) {

cout << "Адрес этого узла : " << currNode << endl;

cout << "Адрес предыдущего узла : " << currNode->prev << endl;

cout << "Адрес следующего узла : " << currNode->next << endl << endl;

print(currNode->Data);

if (currNode->next == NULL) break;

currNode = currNode->next;

}

}

// Очистка списка

void erase\_root(wallpaperNode\* node) {

if (node->next != NULL)

erase\_root(node->next);

if (node->prev != NULL)

{

node->prev->next = NULL;

delete node;

}

}

# Заголовочный файл wallpaper.h

#pragma once

#ifndef WALLPAPER\_H

#define WALLPAPER\_H

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

struct wallpaper {

int length; //Длина рулона

int width; //Ширина рулона

string picture; //Есть ли рисунок на обоях

string canvas; //Тип полотна

string date; //Дата изготовления

int expiration\_date; //Срок годности

string density; //Плотность

float cost; //Цена за рулон

};

struct wallpaperNode {

wallpaper Data;

wallpaperNode\* next = NULL;

wallpaperNode\* prev = NULL;

};

static string database = "Wallpaper.dat";

static string txt\_table = "wallpaper.txt";

static string csv\_table = "wallpaper.csv";

void input\_data();

void output\_data();

void min\_cost();

int table();

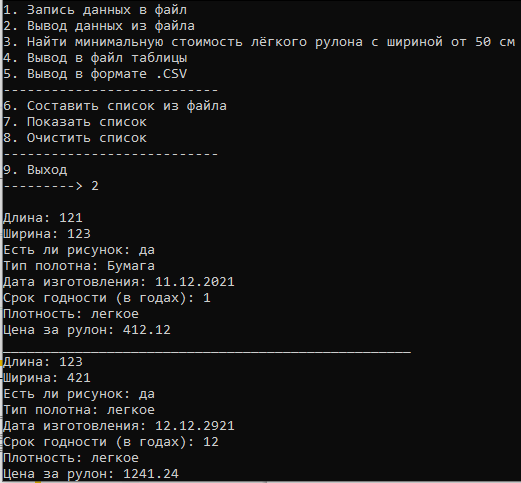
int export\_data();

void create\_root(wallpaperNode\* node);

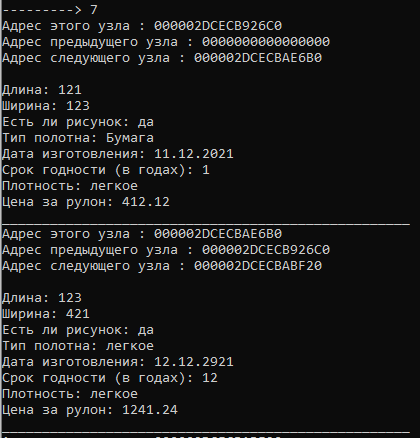
void erase\_root(wallpaperNode\* node);

void show\_root(wallpaperNode\* node);

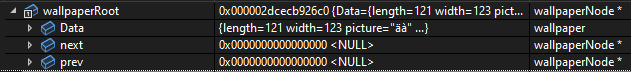
#endif



6 Данные в файле



7 Вывод динамического списка



8 Последний элемент динамического списка