МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
СЕВЕРО-КАВКАЗСКОГО ФЕДЕРАЛЬНОГО УНИВЕРСИТЕТА

ИНСТИТУТА ЦИФРОГО РАЗВИТИЯ

**ОТЧЕТ**

по лабораторной работе №1

**Динамические структуры: стеки, очереди и списки**

Студент: Борсов Б.М группы ПИЖ-б-о-23-1

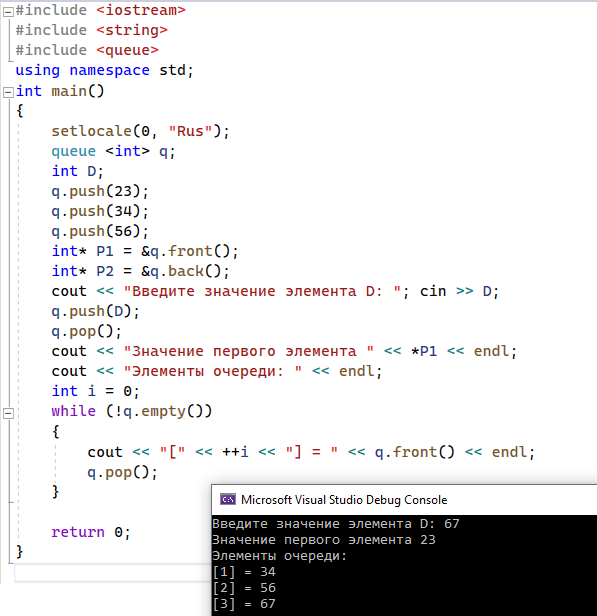
Руководитель практики: Новикова Е.Н.

**Лабораторная работа № 1. Динамические структуры: стеки, очереди и списки**

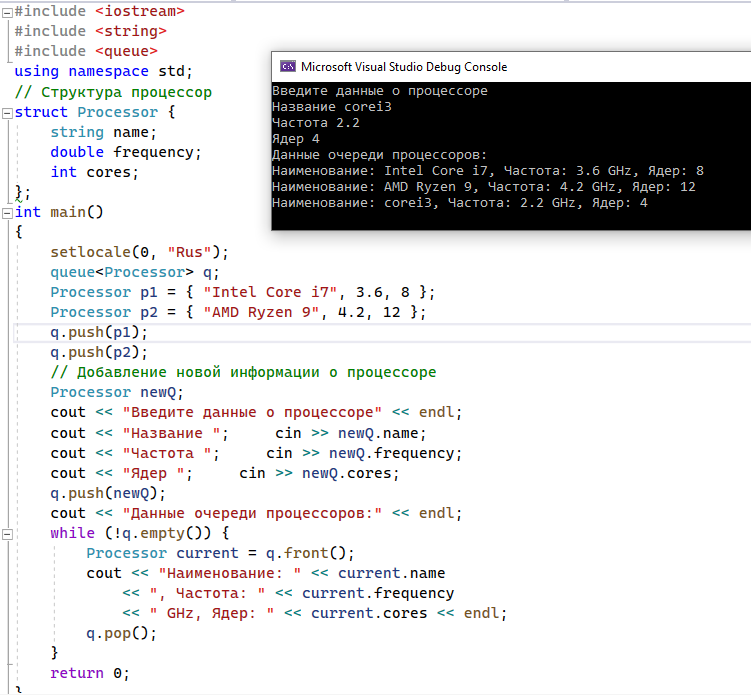
**Цель работы:** освоить программирование задач с динамическими структурами (стеки, очереди, списки)5 варинат

**Задание №1** .

1. Дано число D и указатели P1 и P2 на начало и конец очереди, содержащей не менее двух элементов. Добавить элемент со значением D в конец очереди и извлечь из очереди первый (начальный) элемент. Вывести значение извлеченного элемента и содержимое очереди. После извлечения элемента из очереди освободить память, занимаемую этим элементом.



2. Создать очередь, информационными полями которой являются: наименование процессора и его тактовая частота и количество ядер. Добавить в очередь сведения о новом процессоре. Организовать просмотр данных очереди и распечатать

****

**Задание 2**

1. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 1, 2, 3, 4, 5 и распечатайте содержимое стека. Удалите 3 элемента из стека, и распечатайте содержимое стека еще раз. Найдите сумму элементов стека.

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void push(Node\*& top, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = top;

top = newNode;

}

int pop(Node\*& top) {

if (top == nullptr) {

cout << "Стек пуст!" << endl;

return -1; // Возврат значения по умолчанию для пустого стека

}

int poppedValue = top->data;

Node\* temp = top;

top = top->next;

delete temp;

return poppedValue;

}

// Функция для распечатки содержимого стека

void printStack(Node\* top) {

cout << "Содержимое стека:" << endl;

while (top != nullptr) {

cout << top->data << " ";

top = top->next;

}

cout << endl;

}

// Функция для вычисления суммы элементов стека

int sumStack(Node\* top) {

int sum = 0;

while (top != nullptr) {

sum += top->data;

top = top->next;

}

return sum;

}

int main() {

setlocale(0, "ru");

Node\* top = nullptr;

push(top, 1);

push(top, 2);

push(top, 3);

push(top, 4);

push(top, 5);

// Печать содержимого стека

printStack(top);

// Удаление трех элементов из стека

pop(top);

pop(top);

pop(top);

// Печать содержимого стека после удаления

printStack(top);

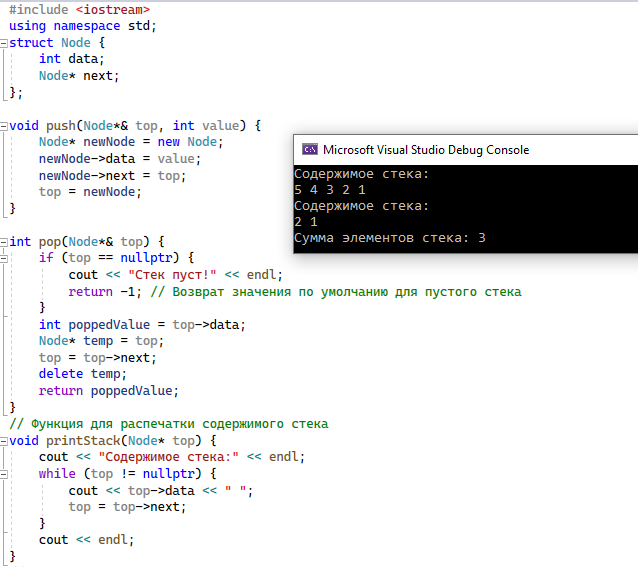
// Вычисление и печать суммы элементов стека

int sum = sumStack(top);

cout << "Сумма элементов стека: " << sum << endl;

return 0;

}



2. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа 46.5, 3.4, 32.4, -3.21 и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь число 5.0 и распечатайте очередь еще раз. Найдите сумму элементов, по модулю больших 12, принадлежащих очереди.

#include <iostream>

#include <cmath>

using namespace std;

// Структура для узла списка

struct Node {

double data;

Node\* next;

};

// Структура для представления очереди

struct Queue {

Node\* front;

Node\* rear;

};

// Функция для инициализации очереди

void initQueue(Queue& q) {

q.front = nullptr;

q.rear = nullptr;

}

// Функция для проверки пустоты очереди

bool isEmpty(const Queue& q) {

return (q.front == nullptr);

}

// Функция для добавления элемента в очередь

void enqueue(Queue& q, double value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (isEmpty(q)) {

q.front = newNode;

q.rear = newNode;

}

else {

q.rear->next = newNode;

q.rear = newNode;

}

}

// Функция для удаления элемента из очереди

double dequeue(Queue& q) {

if (isEmpty(q)) {

cout << "Очередь пуста!" << endl;

return 0; // Возврат значения по умолчанию для пустой очереди

}

double dequeuedValue = q.front->data;

Node\* temp = q.front;

q.front = q.front->next;

if (q.front == nullptr) {

q.rear = nullptr;

}

delete temp;

return dequeuedValue;

}

// Функция для распечатки содержимого очереди

void printQueue(const Queue& q) {

cout << "Содержимое очереди:" << endl;

Node\* current = q.front;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для вычисления суммы элементов, по модулю больших 12

double s(const Queue& q) {

double sum = 0.0;

Node\* current = q.front;

while (current != nullptr) {

if (fabs(current->data) > 12) {

sum += fabs(current->data);

}

current = current->next;

}

return sum;

}

int main() {

setlocale(0 ,"ru");

Queue q;

initQueue(q);

// Добавление элементов в очередь

enqueue(q, 46.5);

enqueue(q, 3.4);

enqueue(q, 32.4);

enqueue(q, -3.21);

// Печать содержимого очереди

printQueue(q);

// Удаление двух элементов из очереди

dequeue(q);

dequeue(q);

// Добавление числа 5.0 в очередь

enqueue(q, 5.0);

// Печать содержимого очереди после изменений

printQueue(q);

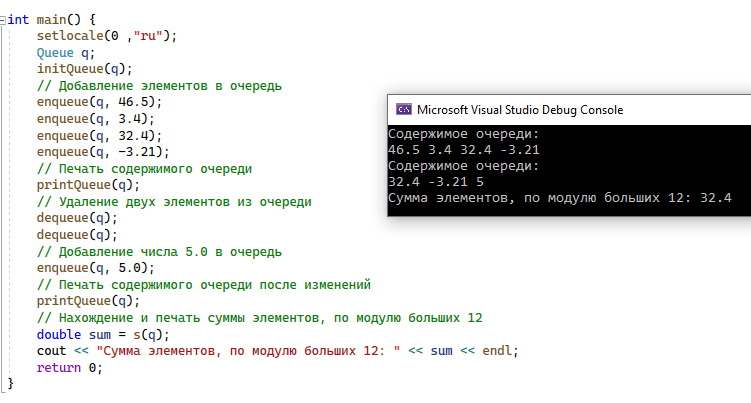
// Нахождение и печать суммы элементов, по модулю больших 12

double sum = s(q);

cout << "Сумма элементов, по модулю больших 12: " << sum << endl;

return 0;

}



**Задание 3**

Дано число N (> 0) и две непустые очереди; адреса начала и конца первой равны P1 и P2, а второй — P3 и P4. Переместить N начальных элементов первой очереди в конец второй очереди. Если первая очередь содержит менее N элементов, то переместить из первой очереди во вторую все элементы. Вывести новые адреса начала и конца первой, а затем второй очереди (для пустой очереди дважды вывести nil). Операции выделения и освобождения памяти не использовать.

#include <iostream>

#include <queue>

using namespace std;

// Функция для перемещения элементов из одной очереди в другую

void moveElements(queue<int>& q1, queue<int>& q2, int N) {

// Если первая очередь пуста, выводим nil в качестве адресов начала и конца

if (q1.empty()) {

cout << " первая очередь пуста, выводим nil в качестве адресов начала и конца" << endl;

cout << "nil nil" << endl;

}

else {

// Сохраняем текущий размер первой очереди

int size = q1.size();

// Перемещаем N или все элементы из первой очереди во вторую

for (int i = 0; i < N && i < size; ++i) {

q2.push(q1.front());

q1.pop();

}

// Если в первой очереди остались элементы, выводим их адресы начала и конца

if (!q1.empty()) {

cout << "в первой очереди остались элементы, вывод адреса начала и конца первой очереди: " << endl;

cout << "&" << q1.front() << " &" << q1.back() << endl;

}

else {

cout << " в первой очереди остались элементы, выводим их адресы начала и конца" << endl;

cout << "nil nil" << endl; // Если первая очередь стала пустой

}

}

// Если вторая очередь пуста, выводим nil в качестве адресов начала и конца

if (q2.empty()) {

cout << "вторая очередь пуста, выводим nil в качестве адресов начала и конца " << endl;

cout << "nil nil" << endl;

}

else {

cout << "Вторая очередь пуста, вывод адреса начала и конца второй очереди: "<< endl;

cout << "&" << q2.front() << " &" << q2.back() << endl;

}

}

int main() {

setlocale(0, "ru");

int N;

cout << "Введите число N (> 0): ";

cin >> N;

queue<int> q1, q2;

// Заполняем первую очередь тестовыми данными

q1.push(1);

q1.push(2);

q1.push(3);

q1.push(4);

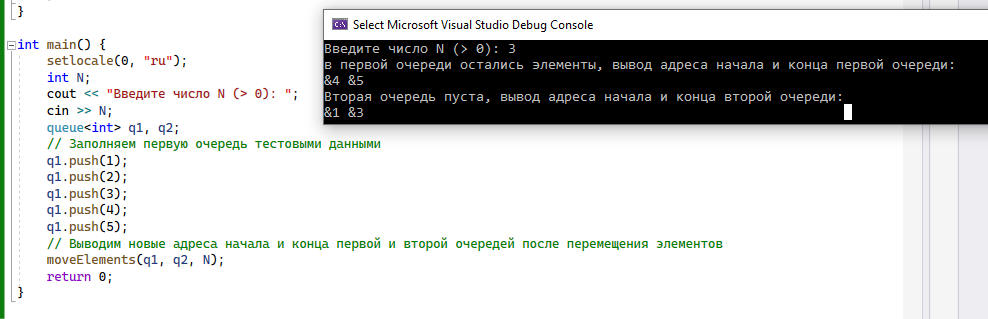
q1.push(5);

// Выводим новые адреса начала и конца первой и второй очередей после перемещения элементов

moveElements(q1, q2, N);

return 0;

}



**Задание 4**

1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент после первого элемента с положительным значением.

#include <iostream>

using namespace std;

// Структура для узла списка

struct Node {

double data;

Node\* next;

};

// Функция для добавления элемента в список

void insert(Node\*& head, double value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

}

else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для удаления элемента после первого положительного элемента

void remove(Node\*& head) {

Node\* current = head;

while (current != nullptr && current->data <= 0) {

current = current->next;

}

if (current != nullptr && current->next != nullptr) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

}

}

// Функция для печати содержимого списка

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для освобождения памяти, занятой списком

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

int main() {

setlocale(0, "ru");

Node\* head = nullptr;

// Добавление элементов в список

insert(head, -5.2);

insert(head, 3.8);

insert(head, -2.7);

insert(head, 7.1);

insert(head, 1.5);

cout << "Cодержимого списка до удаления:" << endl;

printList(head);

// Удаление элемента после первого положительного элемента

remove(head);

cout << "Cодержимого списка после удаления:" << endl;

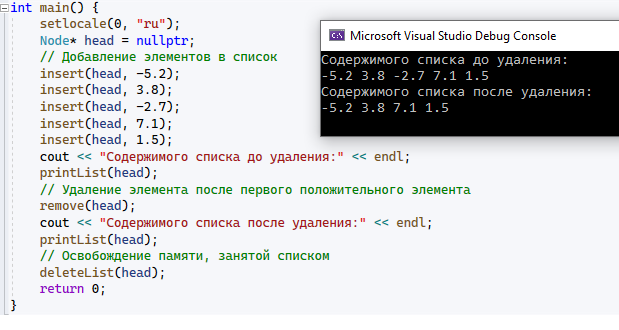
printList(head);

// Освобождение памяти, занятой списком

deleteList(head);

return 0;

}



2. Создать линейный однонаправленный список из целых чисел. Определить среднее арифметическое значений всех элементов списка, кроме второго и третьего.

#include <iostream>

using namespace std;

// Структура для узла списка

struct Node {

int data;

Node\* next;

};

// Функция для добавления элемента в список

void insert(Node\*& head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

}

else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для вычисления среднего арифметического значений всех элементов списка, кроме второго и третьего

double average(Node\* head) {

if (head == nullptr || head->next == nullptr || head->next->next == nullptr) {

cout << "Список содержит недостаточно элементов!" << endl;

return 0.0;

}

Node\* current = head->next->next->next; // Пропускаем первые три элемента

int sum = 0;

int count = 0;

while (current != nullptr) {

sum += current->data;

count++;

current = current->next;

}

return static\_cast<double>(sum) / count;

}

// Функция для печати содержимого списка

void printList(Node\* head) {

cout << "Содержимое списка:" << endl;

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для освобождения памяти, занятой списком

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

int main() {

Node\* head = nullptr;

// Добавление элементов в список

insert(head, 10);

insert(head, 20);

insert(head, 30);

insert(head, 40);

insert(head, 50);

// Печать содержимого списка

printList(head);

// Вычисление и печать среднего арифметического значений всех элементов списка, кроме второго и третьего

double aver = average(head);

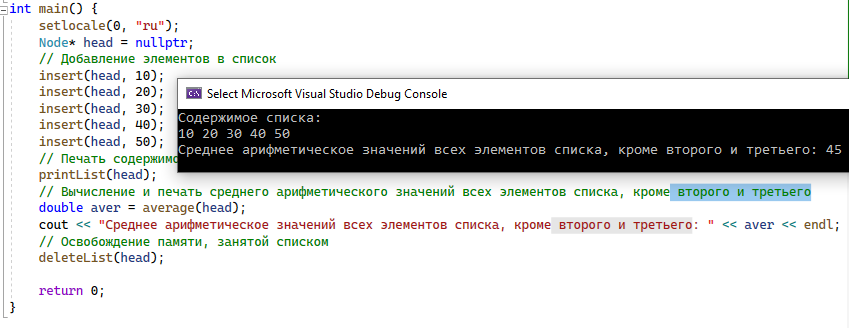
cout << "Среднее арифметическое значений всех элементов списка, кроме второго и третьего: " << aver << endl;

// Освобождение памяти, занятой списком

deleteList(head);

return 0;

}



**Задание 5**

1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка первый элемент меньший по модулю 5.

#include <iostream>

#include <cmath>

using namespace std;

// Структура для узла списка

struct Node {

double data;

Node\* next;

};

// Функция для добавления элемента в список

void insert(Node\*& head, double value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

}

else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для удаления первого элемента, модуль которого меньше 5

void removeFirst5(Node\*& head) {

if (head == nullptr) {

return;

}

if (fabs(head->data) < 5) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* current = head;

while (current->next != nullptr && fabs(current->next->data) >= 5) {

current = current->next;

}

if (current->next != nullptr) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

}

}

// Функция для печати содержимого списка

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для освобождения памяти, занятой списком

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

int main() {

setlocale(0, "ru");

Node\* head = nullptr;

// Добавление элементов в список

insert(head, -3.2);

insert(head, 2.8);

insert(head, -6.1);

insert(head, 8.5);

insert(head, -4.9);

cout << "Содержимое списка до удаления: " << endl;

printList(head);

// Удаление первого элемента, модуль которого меньше 5

removeFirst5(head);

cout << "Содержимое списка после удаления: " << endl;

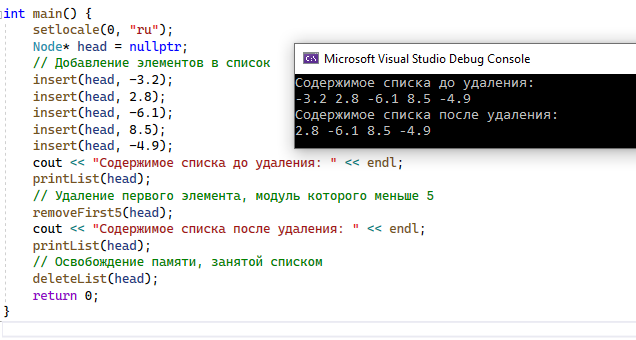
printList(head);

// Освобождение памяти, занятой списком

deleteList(head);

return 0;

}



2. Создать линейный однонаправленный список из вещественных чисел. Определить среднее значение элементов списка со значениями меньшелибо равными 15. Удалить из списка элементы, которые больше 25

#include <iostream>

using namespace std;

// Структура для узла списка

struct Node {

double data;

Node\* next;

};

// Функция для добавления элемента в список

void insert(Node\*& head, double value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

}

else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для вычисления среднего значения элементов списка со значениями меньше или равными 15

double average(Node\* head) {

int sum = 0;

int count = 0;

Node\* current = head;

while (current != nullptr) {

if (current->data <= 15) {

sum += current->data;

count++;

}

current = current->next;

}

if (count == 0) {

return 0.0;

}

return static\_cast<double>(sum) / count;

}

// Функция для удаления элементов списка, которые больше 25

void remove(Node\*& head) {

while (head != nullptr && head->data > 25) {

Node\* temp = head;

head = head->next;

delete temp;

}

if (head == nullptr) {

return;

}

Node\* current = head;

while (current->next != nullptr) {

if (current->next->data > 25) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

}

else {

current = current->next;

}

}

}

// Функция для печати содержимого списка

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция для освобождения памяти, занятой списком

void deleteList(Node\*& head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

int main() {

setlocale(0, "ru");

Node\* head = nullptr;

// Добавление элементов в список

insert(head, 10.5);

insert(head, 20.8);

insert(head, 30.1);

insert(head, 26.3);

insert(head, 5.7);

insert(head, 24.9);

insert(head, 18.2);

cout << "Содержимое списка:" << endl;

printList(head);

// Вычисление и печать среднего значения элементов списка со значениями меньше или равными 15

double av = average(head);

cout << "Среднее значение элементов списка со значениями меньше или равными 15: " << av << endl;

// Удаление элементов списка, которые больше 25

remove(head);

cout << "Содержимое списка после удаления: " << endl;

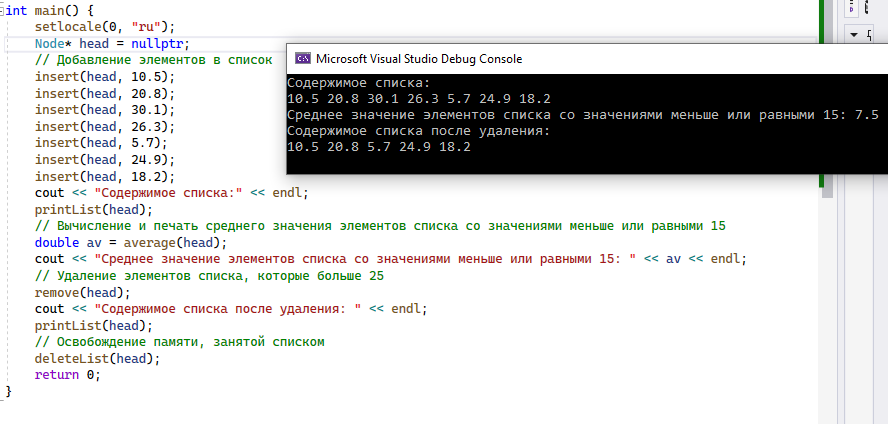
printList(head);

// Освобождение памяти, занятой списком

deleteList(head);

return 0;

}



**Задание 6**

Дан указатель P1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными значениями (новые элементы добавлять перед существующими элементами с такими же значениями) и вывести указатель на первый элемент преобразованного списка.