Банкузов Михаил 7 группа Лабораторная работ №6 Вариант 1



#include <iostream>

#include <fstream>

using namespace std;

// Определение структуры узла

struct Node {

int data;// данные узла

Node\* next;// указатель на следующий узел

};

Node\* head = NULL;// указатель на голову списка

// Функция для добавления узла в конец списка

void insertNode(int value) {

Node\* newNode = new Node(); // выделение памяти под новый узел

newNode->data = value; // инициализация данных узла

newNode->next = NULL; // инициализация указателя на следующий узел

// Если список пустой, то новый узел становится головой списка

if (head == NULL) {

head = newNode;

}

// Иначе находим последний узел списка и добавляем новый узел в конец

else {

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для удаления узла со значением value

void deleteNode(int value) {

// Если список пуст

if (head == NULL) {

cout << "List is empty." << endl;

}

// Если головной узел содержит значение value, удаляем его

else if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

}

// Иначе находим узел со значением value и удаляем его

else {

Node\* current = head;

while (current->next != NULL && current->next->data != value) {

current = current->next;

}

// Если узел со значением value не найден

if (current->next == NULL) {

cout << "Element not found." << endl;

}

// Иначе удаляем найденный узел

else {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

}

}

}

// Функция для вывода списка на экран

void displayList() {

// Если список пустой

if (head == NULL) {

cout << "List is empty." << endl;

}

// Иначе проходим по всем узлам списка и выводим их значения

else {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

cout << endl;

}

}

// Функция для записи списка в файл

void writeToFile() {

ofstream file("list.txt");// открываем файл для записи

// Если файл успешно открыт

if (file.is\_open()) {

Node\* current = head;

while (current != NULL) {

file << current->data << endl;

current = current->next;

}

file.close();// закрываем файл

cout << "List written to file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция чтения списка из файла

void readFromFile() {

ifstream file("list.txt");// открываем файл для чтения

if (file.is\_open()) {

int value;

while (file >> value) {

insertNode(value);

}

file.close();// закрываем файл

cout << "List read from file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция подсчета положительных элементов списка

int sumOfPositiveElements() {

int sum = 0;

Node\* current = head;

while (current != NULL) {

// Если элемент положительный складываем с предыдущим

if (current->data > 0) {

sum += current->data;

}

current = current->next;

}

return sum;

}

int main() {

// Меню

int choice, value;

cout << "Menu:" << endl;

cout << "1. Add element" << endl;

cout << "2. Delete element" << endl;

cout << "3. Display list" << endl;

cout << "4. Write to file" << endl;

cout << "5. Read from file" << endl;

cout << "6. Calculate" << endl;

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to add: ";

cin >> value;

insertNode(value);

main();

break;

case 2:

cout << "Enter value to delete: ";

cin >> value;

deleteNode(value);

main();

break;

case 3:

displayList();

main();

break;

case 4:

writeToFile();

main();

break;

case 5:

readFromFile();

main();

break;

case 6:

int sum = sumOfPositiveElements();

cout << "Sum of positive elements: " << sum << endl;

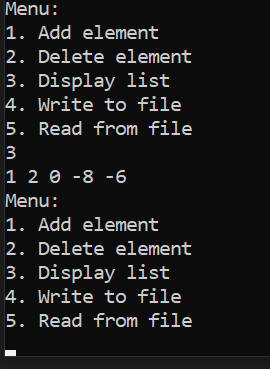
main();

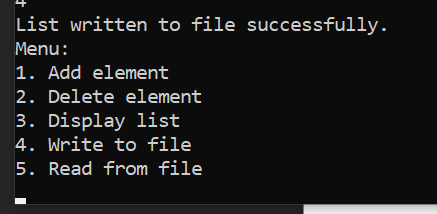
break;

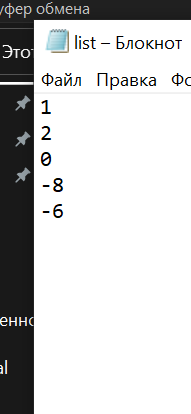
}

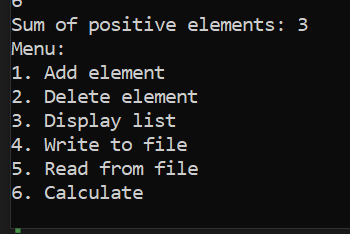
return 0;

}









Дополнительные задания

Вариант 2



#include <iostream>

#include <fstream>

using namespace std;

// Определение структуры узла

struct Node {

double data;// данные узла

Node\* next;// указатель на следующий узел

};

Node\* head = NULL;// указатель на голову списка

// Функция для добавления узла в конец списка

void insertNode(int value) {

Node\* newNode = new Node(); // выделение памяти под новый узел

newNode->data = value; // инициализация данных узла

newNode->next = NULL; // инициализация указателя на следующий узел

// Если список пустой, то новый узел становится головой списка

if (head == NULL) {

head = newNode;

}

// Иначе находим последний узел списка и добавляем новый узел в конец

else {

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для удаления узла со значением value

void deleteNode(int value) {

// Если список пуст

if (head == NULL) {

cout << "List is empty." << endl;

}

// Если головной узел содержит значение value, удаляем его

else if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

}

// Иначе находим узел со значением value и удаляем его

else {

Node\* current = head;

while (current->next != NULL && current->next->data != value) {

current = current->next;

}

// Если узел со значением value не найден

if (current->next == NULL) {

cout << "Element not found." << endl;

}

// Иначе удаляем найденный узел

else {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

}

}

}

// Функция для вывода списка на экран

void displayList() {

// Если список пустой

if (head == NULL) {

cout << "List is empty." << endl;

}

// Иначе проходим по всем узлам списка и выводим их значения

else {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

cout << endl;

}

}

// Функция для записи списка в файл

void writeToFile() {

ofstream file("list.txt");// открываем файл для записи

// Если файл успешно открыт

if (file.is\_open()) {

Node\* current = head;

while (current != NULL) {

file << current->data << endl;

current = current->next;

}

file.close();// закрываем файл

cout << "List written to file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция чтения списка из файла

void readFromFile() {

ifstream file("list.txt");// открываем файл для чтения

if (file.is\_open()) {

int value;

while (file >> value) {

insertNode(value);

}

file.close();// закрываем файл

cout << "List read from file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция для нахождения среднего арифметического положительных элементов списка

double averageOfPositiveElements() {

double sum = 0;

int count = 0;

Node\* current = head;

while (current != NULL) {

if (current->data > 0) {

sum += current->data;

count++;

}

current = current->next;

}

return sum / count;

}

int main() {

int choice;

double value;

cout << "Menu:" << endl;

cout << "1. Add element" << endl;

cout << "2. Delete element" << endl;

cout << "3. Display list" << endl;

cout << "4. Write to file" << endl;

cout << "5. Read from file" << endl;

cout << "6. Calculate" << endl;

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to add: ";

cin >> value;

insertNode(value);

main();

break;

case 2:

cout << "Enter value to delete: ";

cin >> value;

deleteNode(value);

main();

break;

case 3:

displayList();

main();

break;

case 4:

writeToFile();

main();

break;

case 5:

readFromFile();

main();

break;

case 6:

double average = averageOfPositiveElements();

if (isnan(average)) {

cout << "No positive elements in list." << endl;

}

else {

cout << "Average of positive elements: " << average << endl;

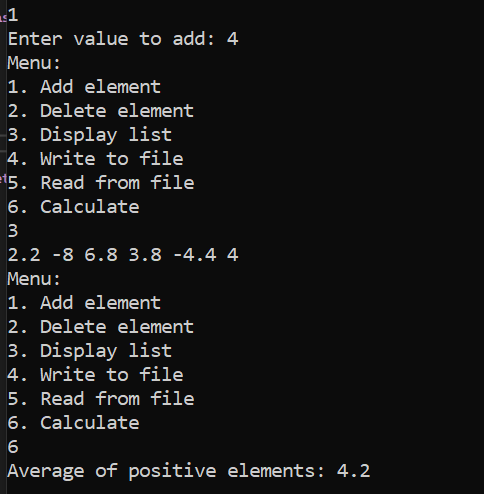
}

}

return 0;

}

Код для записи и чтения из файла остался без изменений



Вариант 3



#include <iostream>

#include <fstream>

using namespace std;

// Определение структуры узла

struct Node {

int data;// данные узла

Node\* next;// указатель на следующий узел

};

Node\* head = NULL;// указатель на голову списка

// Функция для добавления узла в конец списка

void insertNode(int value) {

Node\* newNode = new Node(); // выделение памяти под новый узел

newNode->data = value; // инициализация данных узла

newNode->next = NULL; // инициализация указателя на следующий узел

// Если список пустой, то новый узел становится головой списка

if (head == NULL) {

head = newNode;

}

// Иначе находим последний узел списка и добавляем новый узел в конец

else {

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

// Функция для удаления узла со значением value

void deleteNode(int value) {

// Если список пуст

if (head == NULL) {

cout << "List is empty." << endl;

}

// Если головной узел содержит значение value, удаляем его

else if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

}

// Иначе находим узел со значением value и удаляем его

else {

Node\* current = head;

while (current->next != NULL && current->next->data != value) {

current = current->next;

}

// Если узел со значением value не найден

if (current->next == NULL) {

cout << "Element not found." << endl;

}

// Иначе удаляем найденный узел

else {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

}

}

}

// Функция для вывода списка на экран

void displayList() {

// Если список пустой

if (head == NULL) {

cout << "List is empty." << endl;

}

// Иначе проходим по всем узлам списка и выводим их значения

else {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

cout << endl;

}

}

// Функция для записи списка в файл

void writeToFile() {

ofstream file("list.txt");// открываем файл для записи

// Если файл успешно открыт

if (file.is\_open()) {

Node\* current = head;

while (current != NULL) {

file << current->data << endl;

current = current->next;

}

file.close();// закрываем файл

cout << "List written to file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция чтения списка из файла

void readFromFile() {

ifstream file("list.txt");// открываем файл для чтения

if (file.is\_open()) {

int value;

while (file >> value) {

insertNode(value);

}

file.close();// закрываем файл

cout << "List read from file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

// Функция подсчета положительных элементов списка кратных 5

int sumOfPositiveMultiplesOfFive() {

int sum = 0;

Node\* current = head;

while (current != NULL) {

// Если элемент положительный и кратен 5, то складываем с предыдущим

if (current->data > 0 && current->data % 5 == 0) {

sum += current->data;

}

current = current->next;

}

return sum;

}

int main() {

// Меню

int choice, value;

cout << "Menu:" << endl;

cout << "1. Add element" << endl;

cout << "2. Delete element" << endl;

cout << "3. Display list" << endl;

cout << "4. Write to file" << endl;

cout << "5. Read from file" << endl;

cout << "6. Calculate" << endl;

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to add: ";

cin >> value;

insertNode(value);

main();

break;

case 2:

cout << "Enter value to delete: ";

cin >> value;

deleteNode(value);

main();

break;

case 3:

displayList();

main();

break;

case 4:

writeToFile();

main();

break;

case 5:

readFromFile();

main();

break;

case 6:

int sum = sumOfPositiveMultiplesOfFive();

cout << "Sum of positive multiples of 5: " << sum << endl;

main();

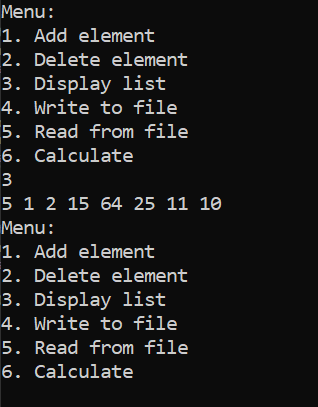
break;

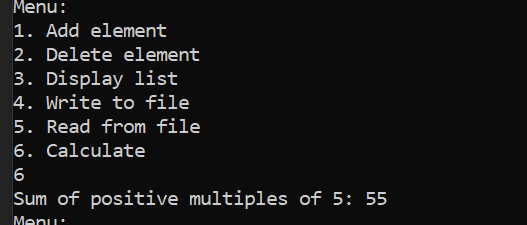
}

return 0;

}

Код для записи и чтения из файла не изменился





Вариант 5



#include <iostream>

#include <fstream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void insertNode(int value) {

Node\* newNode = new Node();

newNode->data = value;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

}

else {

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void deleteNode(int value) {

if (head == NULL) {

cout << "List is empty." << endl;

}

else if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

}

else {

Node\* current = head;

while (current->next != NULL && current->next->data != value) {

current = current->next;

}

if (current->next == NULL) {

cout << "Element not found." << endl;

}

else {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

}

}

}

void displayList() {

if (head == NULL) {

cout << "List is empty." << endl;

}

else {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

cout << endl;

}

}

void writeToFile() {

ofstream file("list.txt");

if (file.is\_open()) {

Node\* current = head;

while (current != NULL) {

file << current->data << endl;

current = current->next;

}

file.close();

cout << "List written to file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

void readFromFile() {

ifstream file("list.txt");

if (file.is\_open()) {

int value;

while (file >> value) {

insertNode(value);

}

file.close();

cout << "List read from file successfully." << endl;

}

else {

cout << "Unable to open file." << endl;

}

}

int sumOfPositiveTwoDigitElements() {

int sum = 0;

Node\* current = head;

while (current != NULL) {

if (current->data > 9 && current->data < 100) {

sum += current->data;

}

current = current->next;

}

return sum;

}

int main() {

int choice, value;

cout << "Menu:" << endl;

cout << "1. Add element" << endl;

cout << "2. Delete element" << endl;

cout << "3. Display list" << endl;

cout << "4. Write to file" << endl;

cout << "5. Read from file" << endl;

cout << "6. Calculate" << endl;

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to add: ";

cin >> value;

insertNode(value);

main();

break;

case 2:

cout << "Enter value to delete: ";

cin >> value;

deleteNode(value);

main();

break;

case 3:

displayList();

main();

break;

case 4:

writeToFile();

main();

break;

case 5:

readFromFile();

main();

break;

case 6:

cout << "Sum of positive two-digit elements in the list: " << sumOfPositiveTwoDigitElements() << endl;

main();

break;

default:

cout << "Invalid choice." << endl;

main();

break;

}

return 0;

}

