|  |  |  |
| --- | --- | --- |
| 1. Изучить работу с *односвязным списком*, выполнив программу, представленную в правой части. Написать условие задачи.  . | #include <iostream>  using namespace std;  struct Item  {  int info;  Item\* next;  };  void main()  {  Item\* plist = nullptr; Item\* p; int number;  for (;;) {  cout << "Input number ";  cin >> number;  if (!number) break;  p = new Item;  p->info = number;  p->next = plist;  plist = p;  }  p = plist;  while (p)  {  cout << p->info << ' ';  p = p->next;  }  while (plist)  {  p = plist;  plist = plist->next;  delete p;  }  } | Создаем структуру содержащую целое число ,создать односвязный список вводя числа от пользователя до тех пор, пока не будет введён 0.вывести все элемента списка |
| 2. В правой части приведена программа, реализующая работу с односвязным списком с использованием *функций пользователя*.  Выполнив программу, изучить способы передачи параметров в функции. | #include <iostream>  using namespace std;  struct list  {  float number;  list\* next;  };  void insert(list\*&, float);  float del(list\*&, float);  int IsEmpty(list  void printList(list\*);  void menu(void);  void sum7(list\*);  int main()  {  setlocale(LC\_CTYPE, "Russian");  list\* first = NULL;  int choice;  float value;  menu();  cout << " ? ";  cin >> choice;  while (choice != 4)  {  switch (choice)  {  case 1: cout << "Введите число ";  cin >> value;  insert(first, value);  printList(first);  break;  case 2: if (!IsEmpty(first))  {  cout << "Введите удаляемое число ";  cin >> value;  if (del(first, value))  {  cout << "Удалено число " << value << endl;  printList(first);  }  else  cout << "Число не найдено" << endl;  }  else  cout << "Список пуст" << endl;  break;  case 3: sum7(first); break;  default: cout << "Неправильный выбор" << endl;  menu();  break;  }  cout << "? ";  cin >> choice;  }  cout << "Конец" << endl;  return 0;  }  void menu(void)  {  cout << "Сделайте выбор:" << endl;  cout << " 1 - Ввод числа" << endl;  cout << " 2 - Удаление числа" << endl;  cout << " 3 - Вычисление суммы чисел, больших 7" << endl;  cout << " 4 - Выход" << endl;  }  void insert(list\*& p, float value)  {  list\* newP = new list;  if (newP != NULL)  {  newP->number = value;  newP->next = p;  p = newP;  }  else  cout << "Операция добавления не выполнена" << endl;  }  float del(list\*& p, float value)  {  list\* previous, \* current, \* temp;  if (value == p->number)  {  temp = p;  p = p->next;  delete temp; return value;  }  else  {  previous = p;  current = p->next;  while (current != NULL && current->number != value)  {  previous = current;  current = current->next;  }  if (current != NULL)  {  temp = current;  previous->next = current->next;  delete(temp);  return value;  }  }  return 0;  }  int IsEmpty(list\* p) {  return p == NULL;  }  void printList(list\* p) {  if (p == NULL)  cout << "Список пуст" << endl;  else  {  cout << "Список:" << endl;  while (p != NULL)  {  cout << "-->" << p->number;  p = p->next;  }  cout << "-->NULL" << endl;  }  }  void sum7(list\* p) {  float sm = 0;  if (p == NULL)  cout << "Список пуст" << endl;  else  {  while (p != NULL)  {  if (p->number > 7)  sm = sm + (p->number);  p = p->next;  }  cout << "Сумма = " << sm << endl;  }  } |  |
| 3. В правой части приведена программа, реализующая работу с односвязным списком.  Выполнив программу, изучить способы *записи списка в файл* и *считывания* из файла. | #include <iostream>  #include <fstream>  #include <limits>  using namespace std;  // Структура для представления элемента односвязного списка  struct list {  char symbol;  list\* next;  };  // Функция для добавления символа в начало списка  void insert(list\*& p, char value) {  list\* newP = new list;  if (newP != nullptr) {  newP->symbol = value;  newP->next = p;  p = newP;  }  else {  cout << "Операция добавления не выполнена" << endl;  }  }  // Функция для вывода списка  void printList(list\* p) {  if (p == nullptr) {  cout << "Список пуст" << endl;  }  else {  cout << "Список:" << endl;  while (p != nullptr) {  cout << "-->" << p->symbol;  p = p->next;  }  cout << "-->NULL" << endl;  }  }  // Функция для записи списка в файл  void toFile(list\* p) {  list\* temp = p;  list buf;  ofstream frm("mList.dat", ios::binary);  if (frm.fail()) {  cout << "\nОшибка открытия файла";  exit(1);  }  while (temp) {  buf = \*temp;  frm.write((char\*)&buf, sizeof(list));  temp = temp->next;  }  frm.close();  cout << "Список записан в файл mList.dat\n";  }  // Функция для считывания списка из файла  void fromFile(list\*& p) {  list buf, \* first = nullptr;  ifstream frm("mList.dat", ios::binary);  if (frm.fail()) {  cout << "\nОшибка открытия файла";  exit(1);  }  frm.read((char\*)&buf, sizeof(list));  while (!frm.eof()) {  insert(first, buf.symbol);  cout << "-->" << buf.symbol;  frm.read((char\*)&buf, sizeof(list));  }  cout << "-->NULL" << endl;  frm.close();  p = first;  cout << "\nСписок считан из файла mList.dat\n";  }  // Функция для вывода меню  void menu(void) {  cout << "Сделайте выбор:" << endl;  cout << "1 - Ввод символа" << endl;  cout << "2 - Запись списка в файл" << endl;  cout << "3 - Чтение списка из файла" << endl;  cout << "4 - Выход" << endl;  }  int main() {  setlocale(LC\_CTYPE, "Russian");  list\* first = nullptr;  int choice;  char value;  menu(); // вывести меню  cout << " ? ";  cin >> choice;  while (choice != 4) {  switch (choice) {  case 1:  cout << "Введите символ ";  cin >> value;  insert(first, value);  printList(first);  break;  case 2:  toFile(first);  break;  case 3:  fromFile(first);  break;  default:  cout << "Неправильный выбор" << endl;  menu();  break;  }  cout << "? ";  cin >> choice;  }  return 0;  } |  |
| Вариант 10  Создать список, содержащий элементы целого типа. Найти сумму отрицательных элементов, кратных 2, или выдать сообщение, что таких элементов нет. | | |
| #include<iostream>  #include<fstream>  #include<limits>  using namespace std;  struct Node {  int value;  Node\* next;  };  void addNode(Node\*& head, int value) {  Node\* newNode = new Node;  newNode->value = value;  newNode->next = head;  head = newNode;  }  void delNode(Node\*& head, int value) {  Node\* current = head;  Node\* previos = nullptr;  while (current != nullptr && current->value != value) {  previos = current;  current = current->next;  }  if (current == nullptr) {  cout << "Элемент не найден." << endl;  return;  }  if (previos == nullptr) {  head = current->next;  }  else {  previos->next = current->next;  }  delete current;  cout << "Элемент удален." << endl;  }  // Функция для поиска элемента в односвязном списке  Node\* searchNode(Node\* head, int value) {  Node\* current = head;  while (current != nullptr) {  if (current->value == value) {  return current;  }  current = current->next;  }  return nullptr;  }  // Функция для вывода односвязного списка в консольное окно  void displayList(Node\* head) {  Node\* current = head;  while (current != nullptr) {  cout << "Значение: " << current->value << endl;  current = current->next;  }  }  // Функция для записи односвязного списка в файл  void writeListToFile(Node\* head, const string& filename) {  ofstream outFile(filename);  if (outFile.is\_open()) {  Node\* current = head;  while (current != nullptr) {  outFile << current->value << endl;  current = current->next;  }  outFile.close();  cout << "Список записан в файл." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  // Функция для считывания односвязного списка из файла  void readListFromFile(Node\*& head, const string& filename) {  ifstream inFile(filename);  if (inFile.is\_open()) {  int value;  head = nullptr; // очищаем текущий список перед чтением из файла  while (inFile >> value) {  addNode(head, value);  }  inFile.close();  cout << "Список считан из файла." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  // Функция для нахождения суммы отрицательных элементов, кратных 2  void findSumOfNegativeEvenElements(Node\* head) {  Node\* current = head;  int sum = 0;  bool found = false;  while (current != nullptr) {  if (current->value < 0 && current->value % 2 == 0) {  sum += current->value;  found = true;  }  current = current->next;  }  if (found) {  cout << "Сумма отрицательных элементов, кратных 2: " << sum << endl;  }  else {  cout << "Таких элементов нет." << endl;  }  }  void menu() {  Node\* head = nullptr;  while (true) {  cout << "\nМеню:\n";  cout << "1. Добавить элемент\n";  cout << "2. Удалить элемент\n";  cout << "3. Поиск элемента\n";  cout << "4. Вывод списка в консольное окно\n";  cout << "5. Запись списка в файл\n";  cout << "6. Считывание списка из файла\n";  cout << "7. Найти сумму отрицательных элементов, кратных 2\n";  cout << "8. Выход\n";  cout << "Выберите опцию: ";  int choice;  cin >> choice;  switch (choice) {  case 1: {  int value;  cout << "Введите значение: ";  cin >> value;  addNode(head, value);  break;  }  case 2: {  int value;  cout << "Введите значение для удаления: ";  cin >> value;  delNode(head, value);  break;  }  case 3: {  int value;  cout << "Введите значение для поиска: ";  cin >> value;  Node\* foundNode = searchNode(head, value);  if (foundNode) {  cout << "Элемент найден: Значение: " << foundNode->value << endl;  }  else {  cout << "Элемент не найден." << endl;  }  break;  }  case 4:  displayList(head);  break;  case 5: {  string filename;  cout << "Введите имя файла для записи: ";  cin >> filename;  writeListToFile(head, filename);  break;  }  case 6: {  string filename;  cout << "Введите имя файла для чтения: ";  cin >> filename;  readListFromFile(head, filename);  break;  }  case 7:  findSumOfNegativeEvenElements(head);  break;  case 8:  return;  default:  cout << "Неверный выбор. Попробуйте снова." << endl;  break;  }  }  }  int main() {  setlocale(LC\_ALL, "ru");  menu();  return 0;  } | |  |

Дополнительные задания

Вариант 11

|  |  |
| --- | --- |
| Создать список, содержащий символы. Найти символ, равный введенному с клавиатуры, вывести этот символ и предыдущий. | |
| #include <iostream>  #include <fstream>  #include <limits>  using namespace std;  struct Node {  char value;  Node\* next;  };  void addNode(Node\*& head, char value) {  Node\* newNode = new Node;  newNode->value = value;  newNode->next = head;  head = newNode;  }  void deleteNode(Node\*& head, char value) {  Node\* current = head;  Node\* previous = nullptr;  while (current != nullptr && current->value != value) {  previous = current;  current = current->next;  }  if (current == nullptr) {  cout << "Элемент не найден." << endl;  return;  }  if (previous == nullptr) {  head = current->next;  }  else {  previous->next = current->next;  }  delete current;  cout << "Элемент удален." << endl;  }  void findAndPrintNodeAndPrevious(Node\* head, char value) {  Node\* current = head;  Node\* previous = nullptr;  while (current != nullptr) {  if (current->value == value) {  if (previous != nullptr) {  cout << "Предыдущий символ: " << previous->value << ", Найденный символ: " << current->value << endl;  }  else {  cout << "Найденный символ: " << current->value << ", Предыдущего символа нет" << endl;  }  return;  }  previous = current;  current = current->next;  }  cout << "Символ не найден." << endl;  }  void displayList(Node\* head) {  Node\* current = head;  while (current != nullptr) {  cout << "Символ: " << current->value << endl;  current = current->next;  }  }  void writeListToFile(Node\* head, const string& filename) {  ofstream outFile(filename);  if (outFile.is\_open()) {  Node\* current = head;  while (current != nullptr) {  outFile << current->value << endl;  current = current->next;  }  outFile.close();  cout << "Список записан в файл." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void readListFromFile(Node\*& head, const string& filename) {  ifstream inFile(filename);  if (inFile.is\_open()) {  char value;  head = nullptr;  while (inFile >> value) {  addNode(head, value);  }  inFile.close();  cout << "Список считан из файла." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void menu() {  Node\* head = nullptr;  while (true) {  cout << "\nМеню:\n";  cout << "1. Добавить элемент\n";  cout << "2. Удалить элемент\n";  cout << "3. Поиск элемента\n";  cout << "4. Вывод списка в консольное окно\n";  cout << "5. Запись списка в файл\n";  cout << "6. Считывание списка из файла\n";  cout << "7. Найти символ и предыдущий\n";  cout << "8. Выход\n";  cout << "Выберите опцию: ";  int choice;  cin >> choice;  switch (choice) {  case 1: {  char value;  cout << "Введите символ: ";  cin >> value;  addNode(head, value);  break;  }  case 2: {  char value;  cout << "Введите символ для удаления: ";  cin >> value;  deleteNode(head, value);  break;  }  case 3: {  char value;  cout << "Введите символ для поиска: ";  cin >> value;  findAndPrintNodeAndPrevious(head, value);  break;  }  case 4:  displayList(head);  break;  case 5: {  string filename;  cout << "Введите имя файла для записи: ";  cin >> filename;  writeListToFile(head, filename);  break;  }  case 6: {  string filename;  cout << "Введите имя файла для чтения: ";  cin >> filename;  readListFromFile(head, filename);  break;  }  case 7: {  char value;  cout << "Введите символ для поиска: ";  cin >> value;  findAndPrintNodeAndPrevious(head, value);  break;  }  case 8:  return;  default:  cout << "Неверный выбор. Попробуйте снова." << endl;  break;  }  }  }  int main() {  setlocale(LC\_ALL, "ru");  menu();  return 0;  } |  |

**Вариант 12**

|  |  |
| --- | --- |
| Создать список, содержащий элементы вещественного типа. Найти среднее значение положительных элементов. | |
| #include <iostream>  #include <fstream>  #include <limits>  using namespace std;  struct Node {  float value;  Node\* next;  };  void addNode(Node\*& head, float value) {  Node\* newNode = new Node;  newNode->value = value;  newNode->next = head;  head = newNode;  }  void deleteNode(Node\*& head, float value) {  Node\* current = head;  Node\* previous = nullptr;  while (current != nullptr && current->value != value) {  previous = current;  current = current->next;  }  if (current == nullptr) {  cout << "Элемент не найден." << endl;  return;  }  if (previous == nullptr) {  head = current->next;  }  else {  previous->next = current->next;  }  delete current;  cout << "Элемент удален." << endl;  }  Node\* searchNode(Node\* head, float value) {  Node\* current = head;  while (current != nullptr) {  if (current->value == value) {  return current;  }  current = current->next;  }  return nullptr;  }  void displayList(Node\* head) {  Node\* current = head;  while (current != nullptr) {  cout << "Значение: " << current->value << endl;  current = current->next;  }  }  void writeListToFile(Node\* head, const string& filename) {  ofstream outFile(filename);  if (outFile.is\_open()) {  Node\* current = head;  while (current != nullptr) {  outFile << current->value << endl;  current = current->next;  }  outFile.close();  cout << "Список записан в файл." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void readListFromFile(Node\*& head, const string& filename) {  ifstream inFile(filename);  if (inFile.is\_open()) {  float value;  head = nullptr;  while (inFile >> value) {  addNode(head, value);  }  inFile.close();  cout << "Список считан из файла." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void findAverageOfPositiveElements(Node\* head) {  Node\* current = head;  float sum = 0;  int count = 0;  while (current != nullptr) {  if (current->value > 0) {  sum += current->value;  count++;  }  current = current->next;  }  if (count > 0) {  cout << "Среднее значение положительных элементов: " << sum / count << endl;  }  else {  cout << "Положительных элементов нет." << endl;  }  }  void menu() {  Node\* head = nullptr;  while (true) {  cout << "\nМеню:\n";  cout << "1. Добавить элемент\n";  cout << "2. Удалить элемент\n";  cout << "3. Поиск элемента\n";  cout << "4. Вывод списка в консольное окно\n";  cout << "5. Запись списка в файл\n";  cout << "6. Считывание списка из файла\n";  cout << "7. Найти среднее значение положительных элементов\n";  cout << "8. Выход\n";  cout << "Выберите опцию: ";  int choice;  cin >> choice;  switch (choice) {  case 1: {  float value;  cout << "Введите значение: ";  cin >> value;  addNode(head, value);  break;  }  case 2: {  float value;  cout << "Введите значение для удаления: ";  cin >> value;  deleteNode(head, value);  break;  }  case 3: {  float value;  cout << "Введите значение для поиска: ";  cin >> value;  Node\* foundNode = searchNode(head, value);  if (foundNode) {  cout << "Элемент найден: Значение: " << foundNode->value << endl;  }  else {  cout << "Элемент не найден." << endl;  }  break;  }  case 4:  displayList(head);  break;  case 5: {  string filename;  cout << "Введите имя файла для записи: ";  cin >> filename;  writeListToFile(head, filename);  break;  }  case 6: {  string filename;  cout << "Введите имя файла для чтения: ";  cin >> filename;  readListFromFile(head, filename);  break;  }  case 7:  findAverageOfPositiveElements(head);  break;  case 8:  return;  default:  cout << "Неверный выбор. Попробуйте снова." << endl;  break;  }  }  }  int main() {  setlocale(LC\_ALL, "ru");  menu();  return 0;  } |  |

**Вариант 13**

|  |  |
| --- | --- |
| Создать список, содержащий элементы целого типа. Найти сумму отрицательных двухзначных элементов или выдать сообщение, что таких элементов нет | |
| #include <iostream>  #include <fstream>  #include <limits>  using namespace std;  struct Node {  int value;  Node\* next;  };  void addNode(Node\*& head, int value) {  Node\* newNode = new Node;  newNode->value = value;  newNode->next = head;  head = newNode;  }  void deleteNode(Node\*& head, int value) {  Node\* current = head;  Node\* previous = nullptr;  while (current != nullptr && current->value != value) {  previous = current;  current = current->next;  }  if (current == nullptr) {  cout << "Элемент не найден." << endl;  return;  }  if (previous == nullptr) {  head = current->next;  }  else {  previous->next = current->next;  }  delete current;  cout << "Элемент удален." << endl;  }  Node\* searchNode(Node\* head, int value) {  Node\* current = head;  while (current != nullptr) {  if (current->value == value) {  return current;  }  current = current->next;  }  return nullptr;  }  void displayList(Node\* head) {  Node\* current = head;  while (current != nullptr) {  cout << "Значение: " << current->value << endl;  current = current->next;  }  }  void writeListToFile(Node\* head, const string& filename) {  ofstream outFile(filename);  if (outFile.is\_open()) {  Node\* current = head;  while (current != nullptr) {  outFile << current->value << endl;  current = current->next;  }  outFile.close();  cout << "Список записан в файл." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void readListFromFile(Node\*& head, const string& filename) {  ifstream inFile(filename);  if (inFile.is\_open()) {  int value;  head = nullptr;  while (inFile >> value) {  addNode(head, value);  }  inFile.close();  cout << "Список считан из файла." << endl;  }  else {  cout << "Ошибка открытия файла." << endl;  }  }  void findSumOfNegativeTwoDigitElements(Node\* head) {  Node\* current = head;  int sum = 0;  bool found = false;  while (current != nullptr) {  if (current->value < 0 && current->value <= -10 && current->value >= -99) {  sum += current->value;  found = true;  }  current = current->next;  }  if (found) {  cout << "Сумма отрицательных двухзначных элементов: " << sum << endl;  }  else {  cout << "Таких элементов нет." << endl;  }  }  void menu() {  Node\* head = nullptr;  while (true) {  cout << "\nМеню:\n";  cout << "1. Добавить элемент\n";  cout << "2. Удалить элемент\n";  cout << "3. Поиск элемента\n";  cout << "4. Вывод списка в консольное окно\n";  cout << "5. Запись списка в файл\n";  cout << "6. Считывание списка из файла\n";  cout << "7. Найти сумму отрицательных двухзначных элементов\n";  cout << "8. Выход\n";  cout << "Выберите опцию: ";  int choice;  cin >> choice;  switch (choice) {  case 1: {  int value;  cout << "Введите значение: ";  cin >> value;  addNode(head, value);  break;  }  case 2: {  int value;  cout << "Введите значение для удаления: ";  cin >> value;  deleteNode(head, value);  break;  }  case 3: {  int value;  cout << "Введите значение для поиска: ";  cin >> value;  Node\* foundNode = searchNode(head, value);  if (foundNode) {  cout << "Элемент найден: Значение: " << foundNode->value << endl;  }  else {  cout << "Элемент не найден." << endl;  }  break;  }  case 4:  displayList(head);  break;  case 5: {  string filename;  cout << "Введите имя файла для записи: ";  cin >> filename;  writeListToFile(head, filename);  break;  }  case 6: {  string filename;  cout << "Введите имя файла для чтения: ";  cin >> filename;  readListFromFile(head, filename);  break;  }  case 7:  findSumOfNegativeTwoDigitElements(head);  break;  case 8:  return;  default:  cout << "Неверный выбор. Попробуйте снова." << endl;  break;  }  }  }  int main() {  setlocale(LC\_ALL, "ru");  menu();  return 0;  } |  |