Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 6

По дисциплине «Основы алгоритмизации и программирования»

На тему «Динамические структуры данных. Односвязные списки»

Выполнила:

Студентка 1 курса 2 группы

Глухова Д.В.

Преподаватель: асс. Андронова М.В.

2023, Минск

**Вариант 2.**

4. В соответствии со своим вариантом разработать программу с использованием *односвязного списка* по данным, представленным в таблице ниже. Программа должна содержать меню с пунктами: добавление элемента, удаление элемента, поиск элемента, вывод списка в консольное окно, запись списка в файл, считывание списка из файла.

**Создать список, содержащий элементы вещественного типа. Найти среднее значение положительных элементов.**

|  |
| --- |
| Программный код |
| // Создать список, содержащий элементы вещественного типа. Найти среднее значение положительных элементов.  #include <iostream>  #include <fstream>  using namespace std;  struct list  {  float number;  list\* next;  };  // прототипы функций  void insert(list\*&, float); //функция добавления элемента, передается адрес списка и символ, который добавляется  float del(list\*&, float); //функция удаления, передается адрес списка и символ, который удаляется  int IsEmpty(list\*); //функция, которая проверяет, пуст ли список  void printList(list\*); //функция вывода  void menu(void); //функция, показывающая меню  void arifm(list\*); //функция подсчета среднего арифметического  void toFile(list\*& p);  void fromFile(list\*& p);  void find(list\*);  int main()  {  setlocale(LC\_CTYPE, "Russian");  list\* first = NULL;  int choice = 0;  float value;  menu(); // вывести меню  while (choice != 8)  {  cout << "Ваш выбор: ";  cin >> choice;  switch (choice)  {  // кейс для добавления числа  case 1: cout << "Введите число "; // добавить число в список  cin >> value;  insert(first, value);  break;  // кейс для вывода листа  case 2:  printList(first);  break;  // кейс для удаления  case 3: if (!IsEmpty(first)) // проверка  {  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;  // кейс для среднего арифметического  case 4: arifm(first);  break;  // кейс для записи в файл  case 5: toFile(first);  break;  // кейс для чтения из файла  case 6: fromFile(first);  break;  // кейс для поиска  case 7: find(first);  break;    default: cout << "Неправильный выбор" << endl;  menu();  break;  }  }  cout << "Конец" << endl;  return 0;  }  void menu(void) //Вывод меню  {  cout << "Сделайте выбор:" << endl;  cout << " 1 - Ввод числа" << endl;  cout << " 2 - Вывод числа" << endl;  cout << " 3 - Удаление числа" << endl;  cout << " 4 - Вычисление среднего арифметического чисел больше нуля" << endl;  cout << " 5 - Запись в файл" << endl;  cout << " 6 - Вывод из файла" << endl;  cout << " 7 - Поиск элемента" << endl;  cout << " 8 - Выход" << endl;  }  // -> доступ по указателю  void insert(list\*& p, float value) //Добавление числа value в список  {  list\* newP = new list;  if (newP != NULL) //есть ли место?  {  newP->number = value; // присваиваем введенное в главной функции вещественное значение  newP->next = p;  p = newP;  }  else  cout << "Операция добавления не выполнена" << endl;  }  float del(list\*& p, float value) // Удаление числа  {  list\* previous, \* current, \* temp;  if (value == p->number) //удаление предыдущего р, чтобы удалить первый р  {  temp = p;  p = p->next; // отсоединить узел  delete temp; //освободить отсоединенный узел  return value;  }  else  {  previous = p;  current = p->next;  while (current != NULL && current->number != value) // выполняется пока не найдем введенное значение(2 штуки перебрасываются последовательно)  {  previous = current;  current = current->next; // перейти к следующему  }  if (current != NULL)  {  temp = current; // то, что будем удалять  previous->next = current->next; // переадресация  free(temp); // удаление  return value;  }  }  return 0;  }  int IsEmpty(list\* p) //Список пустой? (1-да, 0-нет)  { // эту функцию можно сделать типа bool  return p == NULL;  }  void printList(list\* p) //Вывод списка  {  if (p == NULL)  cout << "Список пуст" << endl;  else  {  cout << "Список:" << endl;  while (p != NULL)  {  cout << "\t" << p->number;  p = p->next;  }  cout << "\tNULL" << endl;  }  }  void arifm(list\* p) // Подсчет среднего арифметического  {  float sm = 0, k = 0;  if (p == NULL) // если указатель на элемент списка = NULL  cout << "Список пуст" << endl;  else  {  while (p != NULL) // пока указатель не равен NULL  {  if (p->number > 0) //  {  sm += (p->number);  k++;  }  p = p->next;  }  cout << "Среднее арифметическое = " << sm / k << endl;  }  }  void toFile(list\*& p)  {  list\* temp = p;  ofstream file1("text.txt"); // открываем файл для записи  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (temp) // пока элемент списка не равен NULL  {  float nn = temp->number; // переменная для записи  file1 << nn << "\n"; // записываем  temp = temp->next; // переходим на следующий  }  file1.close(); // закрываем файл  cout << "Список записан в файл text.txt\n";  }  void fromFile(list\*& p)  {  char buf[10]; // буфер  ifstream file1("text.txt"); // открываем файл для чтения  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (!file1.eof()) //пока не дочитали до конца файла  {  file1.getline(buf, 10); //считываем строчку со значением  if (strlen(buf)) //если длина > 0, то  insert(p, atoi(buf)); //добавляем в список  cout << "\t" << buf;  }  cout << "\tNULL" << endl;  file1.close();  }  void find(list\* p) // для поиска элементов  {  bool isFind = false;  int element;  cout << "Введите элемент" << endl;  cin >> element;  list\* find = p;//с начала  while (find) // пока элементы списка не равны NULL  {  if (find->number == element) // если элемент совпадает с введенным  {  isFind = true;//нашли  printf("Элемент найден\n");  break;  }  find = find->next; // иначе перебираем значения  }  if (!isFind)  printf("Элемент не найден\n");  } |
| Консоль отладки |
|  |
| Конечный файл |
|  |

Дополнительные задания.

Вариант 13

|  |
| --- |
| Программный код |
| // Создать список, содержащий элементы целого типа. Найти сумму отрицательных двухзначных элементов или выдать сообщение, что таких элементов нет.  #include <iostream>  #include <fstream>  using namespace std;  struct list  {  int number;  list\* next;  };  // прототипы функций  void insert(list\*&, int); //функция добавления элемента, передается адрес списка и символ, который добавляется  float del(list\*&, int); //функция удаления, передается адрес списка и символ, который удаляется  int IsEmpty(list\*); //функция, которая проверяет, пуст ли список  void printList(list\*); //функция вывода  void menu(void); //функция, показывающая меню  void arifm(list\*); //функция суммы  void toFile(list\*& p);  void fromFile(list\*& p);  void find(list\*);  int main()  {  setlocale(LC\_CTYPE, "Russian");  list\* first = NULL;  int choice = 0;  int value;  menu(); // вывести меню  while (choice != 8)  {  cout << "Ваш выбор: ";  cin >> choice;  switch (choice)  {  // кейс для добавления числа  case 1: cout << "Введите число "; // добавить число в список  cin >> value;  insert(first, value);  break;  // кейс для вывода листа  case 2:  printList(first);  break;  // кейс для удаления  case 3: if (!IsEmpty(first)) // проверка  {  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;  // кейс для суммы  case 4: arifm(first); // вычисление суммы  break;  // кейс для записи в файл  case 5: toFile(first);  break;  // кейс для чтения из файла  case 6: fromFile(first);  break;  // кейс для поиска  case 7: find(first);  break;  default: cout << "Неправильный выбор" << endl;  menu();  break;  }  }  cout << "Конец" << endl;  return 0;  }  void menu(void) //Вывод меню  {  cout << "Сделайте выбор:" << endl;  cout << " 1 - Ввод числа" << endl;  cout << " 2 - Вывод числа" << endl;  cout << " 3 - Удаление числа" << endl;  cout << " 4 - Вычисление суммы" << endl;  cout << " 5 - Запись в файл" << endl;  cout << " 6 - Вывод из файла" << endl;  cout << " 7 - Поиск элемента" << endl;  cout << " 8 - Выход" << endl;  }  // -> доступ по указателю  void insert(list\*& p, int value) //Добавление числа value в список  {  list\* newP = new list;  if (newP != NULL) //есть ли место?  {  newP->number = value; // присваиваем введенное в главной функции вещественное значение  newP->next = p;  p = newP;  }  else  cout << "Операция добавления не выполнена" << endl;  }  float del(list\*& p, int value) // Удаление числа  {  list\* previous, \* current, \* temp;  if (value == p->number) //удаление предыдущего р, чтобы удалить первый р  {  temp = p;  p = p->next; // отсоединить узел  delete temp; //освободить отсоединенный узел  return value;  }  else  {  previous = p;  current = p->next;  while (current != NULL && current->number != value) // выполняется пока не найдем введенное значение(2 штуки перебрасываются последовательно)  {  previous = current;  current = current->next; // перейти к следующему  }  if (current != NULL)  {  temp = current; // то, что будем удалять  previous->next = current->next; // переадресация  free(temp); // удаление  return value;  }  }  return 0;  }  int IsEmpty(list\* p) //Список пустой? (1-да, 0-нет)  { // эту функцию можно сделать типа bool  return p == NULL;  }  void printList(list\* p) //Вывод списка  {  if (p == NULL)  cout << "Список пуст" << endl;  else  {  cout << "Список:" << endl;  while (p != NULL)  {  cout << "\t" << p->number;  p = p->next;  }  cout << "\tNULL" << endl;  }  }  void arifm(list\* p) // Подсчет суммы  {  int sm = 0;  if (p == NULL) // если указатель на элемент списка = NULL  cout << "Список пуст" << endl;  else  {  while (p != NULL) // пока указатель не равен NULL  {  if (p->number >= -99 && p->number <= -10 )  {  sm += (p->number);  }  p = p->next;  }  cout << "Сумма отрицательных двузначных = " << sm << endl;  }  }  void toFile(list\*& p)  {  list\* temp = p;  ofstream file1("text.txt"); // открываем файл для записи  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (temp) // пока элемент списка не равен NULL  {  int nn = temp->number; // переменная для записи  file1 << nn << "\n"; // записываем  temp = temp->next; // переходим на следующий  }  file1.close(); // закрываем файл  cout << "Список записан в файл text.txt\n";  }  void fromFile(list\*& p)  {  char buf[10]; // буфер  ifstream file1("text.txt"); // открываем файл для чтения  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (!file1.eof()) //пока не дочитали до конца файла  {  file1.getline(buf, 10); //считываем строчку со значением  if (strlen(buf)) //если длина > 0, то  insert(p, atoi(buf)); //добавляем в список  cout << "\t" << buf;  }  cout << "\tNULL" << endl;  file1.close();  }  void find(list\* p) // для поиска элементов  {  bool isFind = false;  int element;  cout << "Введите элемент" << endl;  cin >> element;  list\* find = p;//с начала  while (find) // пока элементы списка не равны NULL  {  if (find->number == element) // если элемент совпадает с введенным  {  isFind = true;//нашли  printf("Элемент найден\n");  break;  }  find = find->next; // иначе перебираем значения  }  if (!isFind)  printf("Элемент не найден\n");  } |
| Консоль отладки |
|  |
| Конечный файл |
|  |

Вариант 5

Создать список, содержащий элементы целого типа. Найти сумму положительных двухзначных элементов или выдать сообщение, что таких элементов нет.

|  |
| --- |
| Программный код |
| // Создать список, содержащий элементы целого типа.Найти сумму положительных двухзначных элементов или выдать сообщение, что таких элементов нет.  #include <iostream>  #include <fstream>  using namespace std;  struct list  {  int number;  list\* next;  };  // прототипы функций  void insert(list\*&, int); //функция добавления элемента, передается адрес списка и символ, который добавляется  float del(list\*&, int); //функция удаления, передается адрес списка и символ, который удаляется  int IsEmpty(list\*); //функция, которая проверяет, пуст ли список  void printList(list\*); //функция вывода  void menu(void); //функция, показывающая меню  void arifm(list\*); //функция суммы  void toFile(list\*& p);  void fromFile(list\*& p);  void find(list\*);  int main()  {  setlocale(LC\_CTYPE, "Russian");  list\* first = NULL;  int choice = 0;  int value;  menu(); // вывести меню  while (choice != 8)  {  cout << "Ваш выбор: ";  cin >> choice;  switch (choice)  {  // кейс для добавления числа  case 1: cout << "Введите число "; // добавить число в список  cin >> value;  insert(first, value);  break;  // кейс для вывода листа  case 2:  printList(first);  break;  // кейс для удаления  case 3: if (!IsEmpty(first)) // проверка  {  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;  // кейс для суммы  case 4: arifm(first); // вычисление суммы  break;  // кейс для записи в файл  case 5: toFile(first);  break;  // кейс для чтения из файла  case 6: fromFile(first);  break;  // кейс для поиска  case 7: find(first);  break;  default: cout << "Неправильный выбор" << endl;  menu();  break;  }  }  cout << "Конец" << endl;  return 0;  }  void menu(void) //Вывод меню  {  cout << "Сделайте выбор:" << endl;  cout << " 1 - Ввод числа" << endl;  cout << " 2 - Вывод числа" << endl;  cout << " 3 - Удаление числа" << endl;  cout << " 4 - Вычисление суммы" << endl;  cout << " 5 - Запись в файл" << endl;  cout << " 6 - Вывод из файла" << endl;  cout << " 7 - Поиск элемента" << endl;  cout << " 8 - Выход" << endl;  }  // -> доступ по указателю  void insert(list\*& p, int value) //Добавление числа value в список  {  list\* newP = new list;  if (newP != NULL) //есть ли место?  {  newP->number = value; // присваиваем введенное в главной функции вещественное значение  newP->next = p;  p = newP;  }  else  cout << "Операция добавления не выполнена" << endl;  }  float del(list\*& p, int value) // Удаление числа  {  list\* previous, \* current, \* temp;  if (value == p->number) //удаление предыдущего р, чтобы удалить первый р  {  temp = p;  p = p->next; // отсоединить узел  delete temp; //освободить отсоединенный узел  return value;  }  else  {  previous = p;  current = p->next;  while (current != NULL && current->number != value) // выполняется пока не найдем введенное значение(2 штуки перебрасываются последовательно)  {  previous = current;  current = current->next; // перейти к следующему  }  if (current != NULL)  {  temp = current; // то, что будем удалять  previous->next = current->next; // переадресация  free(temp); // удаление  return value;  }  }  return 0;  }  int IsEmpty(list\* p) //Список пустой? (1-да, 0-нет)  { // эту функцию можно сделать типа bool  return p == NULL;  }  void printList(list\* p) //Вывод списка  {  if (p == NULL)  cout << "Список пуст" << endl;  else  {  cout << "Список:" << endl;  while (p != NULL)  {  cout << "\t" << p->number;  p = p->next;  }  cout << "\tNULL" << endl;  }  }  void arifm(list\* p) // Подсчет суммы  {  int sm = 0;  if (p == NULL) // если указатель на элемент списка = NULL  cout << "Список пуст" << endl;  else  {  while (p != NULL) // пока указатель не равен NULL  {  if (p->number <= 99 && p->number >= 10)  {  sm += (p->number);  }  p = p->next;  }  cout << "Сумма положительных двузначных = " << sm << endl;  }  }  void toFile(list\*& p)  {  list\* temp = p;  ofstream file1("text.txt"); // открываем файл для записи  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (temp) // пока элемент списка не равен NULL  {  int nn = temp->number; // переменная для записи  file1 << nn << "\n"; // записываем  temp = temp->next; // переходим на следующий  }  file1.close(); // закрываем файл  cout << "Список записан в файл text.txt\n";  }  void fromFile(list\*& p)  {  char buf[10]; // буфер  ifstream file1("text.txt"); // открываем файл для чтения  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (!file1.eof()) //пока не дочитали до конца файла  {  file1.getline(buf, 10); //считываем строчку со значением  if (strlen(buf)) //если длина > 0, то  insert(p, atoi(buf)); //добавляем в список  cout << "\t" << buf;  }  cout << "\tNULL" << endl;  file1.close();  }  void find(list\* p) // для поиска элементов  {  bool isFind = false;  int element;  cout << "Введите элемент" << endl;  cin >> element;  list\* find = p;//с начала  while (find) // пока элементы списка не равны NULL  {  if (find->number == element) // если элемент совпадает с введенным  {  isFind = true;//нашли  printf("Элемент найден\n");  break;  }  find = find->next; // иначе перебираем значения  }  if (!isFind)  printf("Элемент не найден\n");  } |
| Консоль отладки |
|  |
| Конечный файл |
|  |

Вариант 3

Создать список, содержащий элементы целого типа. Найти сумму положительных элементов, кратных 5, или выдать сообщение, что таких элементов нет.

|  |
| --- |
| Программный код |
| // Создать список, содержащий элементы целого типа. Найти сумму положительных элементов, кратных 5, или выдать сообщение, что таких элементов нет.  #include <iostream>  #include <fstream>  using namespace std;  struct list  {  int number;  list\* next;  };  // прототипы функций  void insert(list\*&, int); //функция добавления элемента, передается адрес списка и символ, который добавляется  float del(list\*&, int); //функция удаления, передается адрес списка и символ, который удаляется  int IsEmpty(list\*); //функция, которая проверяет, пуст ли список  void printList(list\*); //функция вывода  void menu(void); //функция, показывающая меню  void arifm(list\*); //функция суммы  void toFile(list\*& p);  void fromFile(list\*& p);  void find(list\*);  int main()  {  setlocale(LC\_CTYPE, "Russian");  list\* first = NULL;  int choice = 0;  int value;  menu(); // вывести меню  while (choice != 8)  {  cout << "Ваш выбор: ";  cin >> choice;  switch (choice)  {  // кейс для добавления числа  case 1: cout << "Введите число "; // добавить число в список  cin >> value;  insert(first, value);  break;  // кейс для вывода листа  case 2:  printList(first);  break;  // кейс для удаления  case 3: if (!IsEmpty(first)) // проверка  {  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;  // кейс для суммы  case 4: arifm(first); // вычисление суммы  break;  // кейс для записи в файл  case 5: toFile(first);  break;  // кейс для чтения из файла  case 6: fromFile(first);  break;  // кейс для поиска  case 7: find(first);  break;  default: cout << "Неправильный выбор" << endl;  menu();  break;  }  }  cout << "Конец" << endl;  return 0;  }  void menu(void) //Вывод меню  {  cout << "Сделайте выбор:" << endl;  cout << " 1 - Ввод числа" << endl;  cout << " 2 - Вывод числа" << endl;  cout << " 3 - Удаление числа" << endl;  cout << " 4 - Вычисление суммы" << endl;  cout << " 5 - Запись в файл" << endl;  cout << " 6 - Вывод из файла" << endl;  cout << " 7 - Поиск элемента" << endl;  cout << " 8 - Выход" << endl;  }  // -> доступ по указателю  void insert(list\*& p, int value) //Добавление числа value в список  {  list\* newP = new list;  if (newP != NULL) //есть ли место?  {  newP->number = value; // присваиваем введенное в главной функции вещественное значение  newP->next = p;  p = newP;  }  else  cout << "Операция добавления не выполнена" << endl;  }  float del(list\*& p, int value) // Удаление числа  {  list\* previous, \* current, \* temp;  if (value == p->number) //удаление предыдущего р, чтобы удалить первый р  {  temp = p;  p = p->next; // отсоединить узел  delete temp; //освободить отсоединенный узел  return value;  }  else  {  previous = p;  current = p->next;  while (current != NULL && current->number != value) // выполняется пока не найдем введенное значение(2 штуки перебрасываются последовательно)  {  previous = current;  current = current->next; // перейти к следующему  }  if (current != NULL)  {  temp = current; // то, что будем удалять  previous->next = current->next; // переадресация  free(temp); // удаление  return value;  }  }  return 0;  }  int IsEmpty(list\* p) //Список пустой? (1-да, 0-нет)  { // эту функцию можно сделать типа bool  return p == NULL;  }  void printList(list\* p) //Вывод списка  {  if (p == NULL)  cout << "Список пуст" << endl;  else  {  cout << "Список:" << endl;  while (p != NULL)  {  cout << "\t" << p->number;  p = p->next;  }  cout << "\tNULL" << endl;  }  }  void arifm(list\* p) // Подсчет суммы  {  int sm = 0;  if (p == NULL) // если указатель на элемент списка = NULL  cout << "Список пуст" << endl;  else  {  while (p != NULL) // пока указатель не равен NULL  {  if (p->number % 5 == 0)  {  sm += (p->number);  }  p = p->next;  }  cout << "Сумма кратных 5 = " << sm << endl;  }  }  void toFile(list\*& p)  {  list\* temp = p;  ofstream file1("text.txt"); // открываем файл для записи  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (temp) // пока элемент списка не равен NULL  {  int nn = temp->number; // переменная для записи  file1 << nn << "\n"; // записываем  temp = temp->next; // переходим на следующий  }  file1.close(); // закрываем файл  cout << "Список записан в файл text.txt\n";  }  void fromFile(list\*& p)  {  char buf[10]; // буфер  ifstream file1("text.txt"); // открываем файл для чтения  if (file1.fail()) // если файл не открыт  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (!file1.eof()) //пока не дочитали до конца файла  {  file1.getline(buf, 10); //считываем строчку со значением  if (strlen(buf)) //если длина > 0, то  insert(p, atoi(buf)); //добавляем в список  cout << "\t" << buf;  }  cout << "\tNULL" << endl;  file1.close();  }  void find(list\* p) // для поиска элементов  {  bool isFind = false;  int element;  cout << "Введите элемент" << endl;  cin >> element;  list\* find = p;//с начала  while (find) // пока элементы списка не равны NULL  {  if (find->number == element) // если элемент совпадает с введенным  {  isFind = true;//нашли  printf("Элемент найден\n");  break;  }  find = find->next; // иначе перебираем значения  }  if (!isFind)  printf("Элемент не найден\n");  } |
| Консоль отладки |
|  |
| Конечный файл |
|  |