Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра Программной Инженерии

Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двусвязные списки»

Выполнил:

Студент 1 курса 3 группы

Шатерник Г.И.

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 15

2. Дополнить программу функцией в соответствии со своим вариантом из таблицы, представленной ниже.

|  |
| --- |
| Программный код  //библиотеки  #include <iostream>  #include <fstream>  #include<Windows.h>  using namespace std; //пространство имен  const unsigned int NAME\_SIZE = 30; //создаем константу размера длинны имени и города  const unsigned int CITY\_SIZE = 20;  struct Address // создаем структуру типа Address  {  char name[NAME\_SIZE];  char city[CITY\_SIZE];  Address\* next; //создаем указатель на следующий элемент  Address\* prev; // и на предыдущий элемент  };  int menu(void) //меню  {  char s[80]; int c;  cout << endl;  cout << "1. Ввод имени" << endl;  cout << "2. Удаление имени" << endl;  cout << "3. Вывод на экран" << endl;  cout << "4. Поиск" << endl;  cout << "5. Выход" << endl;  cout << "6. Запись в файла" << endl;  cout << "7. Вывод из файла" << endl;  cout << "8. Добавления в конец списка всех элементов некоторого списка L " << endl;  cout << endl;  do  {  cout << "Ваш выбор: ";  cin.sync(); //важный момент- эта строка убирает все непрочитанные символы из входного потока. С ее помощью можно убрать непрочитанные данные, перед тем, как начать чтение следующей порции данных.  gets\_s(s); //чтение входных данных с помощью функции gets\_s  cout << endl;  c = atoi(s);  } while (c < 0 || c > 9);  return c;  }  void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка  {  Address\* p = \*plast;  if (\*plast == NULL) // если указатель пласт не имеет значение в ячейке памяти  {  e->next = NULL; //то следующий и предыдущий элемент присваиваем null  e->prev = NULL;  \*plast = e;  \*phead = e;  return;  }  else  {  p->next = e;  e->next = NULL;  e->prev = p;  \*plast = e;  }  }  Address\* setElement() // Создание элемента и ввод его значений с клавиатуры  {  Address\* temp = new Address();  if (!temp)  {  cerr << "Ошибка выделения памяти памяти";  return NULL;  }  cout << "Введите имя: ";  cin.getline(temp->name, NAME\_SIZE - 1, '\n'); //с помощью getline задаем значение temp->name и уменьшаем NAME\_SIZE чтобы следить за превышением лимита символов для имени  cin.ignore(cin.rdbuf()->in\_avail()); //игнорирование входного потока, чтобы например не образать внимание на перевод на следующую строку с помощью enter  cin.clear(); // очистка потока  cout << "Введите город: ";  cin.getline(temp->city, CITY\_SIZE - 1, '\n'); //аналогичные операции, то и выше  cin.ignore(cin.rdbuf()->in\_avail());  cin.clear();  temp->next = NULL;  temp->prev = NULL;  return temp;  }  void outputList(Address\*\* phead, Address\*\* plast) //Вывод списка на экран  {  Address\* t = \*phead;  while (t)  {  cout << t->name << ' ' << t->city << endl;  t = t->next;  }  cout << "" << endl;  }  void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке  {  Address\* t = \*phead;  while (t)  {  if (!strcmp(name, t->name)) break; //сравниваем строки.  //если name равно t->name, то присваиваем следующее данное значние  t = t->next;  }  if (!t)  cerr << "Имя не найдено" << endl;  else  cout << t->name << ' ' << t->city << endl;  }  void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) // Удаление имени  {  struct Address\* t = \*phead;  while (t)  {  if (!strcmp(name, t->name)) break;  t = t->next;  }  if (!t)  cerr << "Имя не найдено" << endl;  else  {  if (\*phead == t)  {  \*phead = t->next;  if (\*phead)  (\*phead)->prev = NULL;  else  \*plast = NULL;  }  else  {  t->prev->next = t->next;  if (t != \*plast)  t->next->prev = t->prev;  else  \*plast = t->prev;  }  delete t;  cout << "Элемент удален" << endl;  }  }  void writeToFile(Address\*\* phead) //Запись в файл  {  struct Address\* t = \*phead;  FILE\* fp;  errno\_t err = fopen\_s(&fp, "mlist", "wb");  if (err)  {  cerr << "Файл не открывается" << endl;  exit(1);  }  cout << "Сохранение в файл" << endl;  while (t)  {  fwrite(t, sizeof(struct Address), 1, fp);  t = t->next;  }  fclose(fp);  }  void readFromFile(Address\*\* phead, Address\*\* plast) //Считывание из файла  {  struct Address\* t;  FILE\* fp;  errno\_t err = fopen\_s(&fp, "mlist", "rb");  if (err)  {  cerr << "Файл не открывается" << endl;  exit(1);  }  while (\*phead)  {  \*plast = (\*phead)->next;  delete\* phead;  \*phead = \*plast;  }  \*phead = \*plast = NULL;  cout << "Загрузка из файла" << endl;  while (!feof(fp))  {  t = new Address();  if (!t)  {  cerr << "Ошибка выделения памяти" << endl;  return;  }  if (1 != fread(t, sizeof(struct Address), 1, fp)) break;  insert(t, phead, plast);  }  fclose(fp);  }  void Laba(Address\*\* phead, Address\*\* plast, Address\*\* head, Address\*\* last) {  int kol;  cout << "Введите количество элементов нового списка ";  cin >> kol; //запрашиваем количество элементов для вставки в уже существующую структуру  for (int i = 0; i < kol; i++)  {  insert(setElement(), head, plast);  }  Address\* t = \*head;  while (t) {  insert(t, phead, plast); //вставляем элементы в конец списка  t = t->next;  }  }  int main(void)  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  Address\* head = NULL;  Address\* last = NULL;  Address\* head2 = NULL;  Address\* last2 = NULL;  setlocale(LC\_CTYPE, "Rus");  //в участке кода ниже мы используем выше обьявленные функции и передаем в аргументы элементы которые мы хотим использовать для работы со структурой. Все функции и методы, а также их работа были описаны выше.  while (true)  {  switch (menu())  {  case 1: insert(setElement(), &head, &last);  break;  case 2: { char dname[NAME\_SIZE];  cout << "Введите имя: ";  cin.getline(dname, NAME\_SIZE - 1, '\n');  cin.ignore(cin.rdbuf()->in\_avail());  cin.sync();  delet(dname, &head, &last);  break;  }  case 3: outputList(&head, &last);  break;  case 4: { char fname[NAME\_SIZE];  cout << "Введите имя: ";  cin.getline(fname, NAME\_SIZE - 1, '\n');  cin.ignore(cin.rdbuf()->in\_avail());  cin.sync();  find(fname, &head);  break;  }  case 5: exit(0);  case 6: {  writeToFile(&head);  break; }  case 7: {  readFromFile(&head, &last);  break;  }  case 8: {  Laba(&head, &last, &head2, &last2);  break;  }  default: exit(1);  }  }  return 0;  } |
| Вывод: |