Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двусвязные списки»

Выполнила:

Студентка 1 курса 2 группы

Глухова Д.В.

Преподаватель: асс. Андронова М.В.

2023, Минск

**Вариант 2.**

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

**deleteKFirst(int k)** – функция удаления **К** первых элементов списка.

|  |
| --- |
| Программный код |
| #include <iostream>  #include <windows.h>  using namespace std;  const unsigned int NAME\_SIZE = 30;  const unsigned int CITY\_SIZE = 20;  struct Address  {  char name[NAME\_SIZE]; // информационное поле  char city[CITY\_SIZE]; // информационное поле  Address\* next; // указатель на следующий  Address\* prev; // указатель на предыдущий  };  // прототипы функций  void insert(Address\* e, Address\*\* phead, Address\*\* plast);  Address\* setElement();  void outputList(Address\*\* phead, Address\*\* plast);  void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast);  void find(char name[NAME\_SIZE], Address\*\* phead);  void writeToFile(Address\*\* phead);  void readFromFile(Address\*\* phead, Address\*\* plast);  void deleteKFirst(Address\*\* phead, Address\*\* plast);  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_CTYPE, "Rus");  Address\* head = NULL;  Address\* last = NULL;  short choose = 0;  cout << "Выберите действие:\n1 — ввод элемента;\n2 — вывод списка;\n3 — удаление элемента;\n4 — поиск элемента;"  "\n5 — запись в файл;\n6 — вывод из файла;\n7 — удаление первых k элементов;\n8 — выход из программы.\n";  while (choose != 5)  {  char s[80]; int c;  cin.sync();  gets\_s(s);  cout << endl;  c = atoi(s);  switch (c)  {  case 1:  insert(setElement(), &head, &last);  break;  case 2:  outputList(&head, &last);  break;  case 3:  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 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:  writeToFile(&head);  break;  case 6:  readFromFile(&head, &last);  break;  case 7:  deleteKFirst(&head, &last);  break;  case 8:  cout << "До свидания!\n";  exit(0);  }  }  }  void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка  {  Address\* p = \*plast; // указатель на последний  if (\*plast == NULL) // если указывает на пустой элемент, то  {  e->next = NULL;  e->prev = NULL;  \*plast = e; // добавляем  \*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.\n";  return NULL;  }  cout << "Введите имя: ";  // функции для работы со строками для большей корректности  cin.getline(temp->name, NAME\_SIZE - 1, '\n');  cin.ignore(cin.rdbuf()->in\_avail());  cin.clear();  cout << "Введите город: ";  cin.getline(temp->city, CITY\_SIZE - 1, '\n');  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 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.\n" << endl;  else  {  if (\*phead == t)  {  \*phead = t->next;  if (\*phead)  (\*phead)->prev = NULL; // удаление из начала  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.\n" << endl;  }  }  void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке  {  Address\* t = \*phead; // начиная с заголовка  while (t) // пока не конец списка  {  if (!strcmp(name, t->name)) break; // сравниваем строки  t = t->next; // нас следующий  }  if (!t)  cerr << "Имя не найдено.\n" << endl;  else  cout << t->name << ' ' << t->city << endl; // выводим найденный  }  void writeToFile(Address\*\* phead) //Запись в файл  {  struct Address\* t = \*phead;  FILE\* fp;  errno\_t err = fopen\_s(&fp, "mlist.txt", "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 фай-ла  {  struct Address\* t;  FILE\* fp;  errno\_t err = fopen\_s(&fp, "mlist.txt", "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 deleteKFirst(Address\*\* phead, Address\*\* plast)  {  Address\* t = \*phead;  short k, i = 0;  cout << "Введите k:\n";  cin >> k; // количество сиимволов, которые нужно удалить  for (int i = 0; t; i++) // цикл для удаления  {  if (k) // пока true  {  if (t->next != NULL) // если след не равен NULL  {  t = t->next; // перемещаемся  delet(t->prev->name, phead, plast); // вызываем функцию delet для нашего элемента  }  else  {  delet(t->name, phead, plast); // иначе удаляем  t = NULL; // и ставим NULL  }  k--; // уменьшеем количество сиимволов, которые нужно удалить  }  else t = t->next;  }  } |
| Консоль отладки |
|  |

Дополнительные задания.

1. В [приложении 2](file:///E:\уник\лабы\ПрактикумОАП-2часть2018_Пацей.doc#prilogenie2) приведен проект, в котором реализован проект с использованием *двусвязного* *списка*. На основе данного проекта разработать функции, которые предлагается создать в данном приложении.

|  |
| --- |
| Программный код |
| Главный файл |
| #include "List.h"  #include <iostream>  using namespace std;  struct HotelClients  {  string passport;  string fio;  string dateIn;  string dateOut;  string num;  string typeNum;  HotelClients\* next;  };  void print(void\* b) //Функция используется при выводе  {  HotelClients\* a = (HotelClients\*)b;  cout << a->passport << " " << a->fio << " " << a->dateIn << " " << a->dateOut << " " << a->num << " " << a->typeNum << endl;  }  int main()  {  setlocale(LC\_ALL, "Russian");  HotelClients a1 = { "HB1234567", "Петров", "23.03.2023", "29.03.2023", "317", "НЕ ЛЮКС!"};  HotelClients a2 = { "CV6543217", "Сидоров", "25.03.2023", "27.03.2023", "507", "двушка"};  HotelClients a3 = { "MV8765432", "Синицын", "09.03.2023", "26.03.2023", "319", "однушка"};  Object L1 = Create();  L1.Insert(&a3);  L1.Insert(&a2);  L1.Insert(&a1);  bool rc;  int c, n;  do {  cout << "Выберите, что сделать:" << endl;  cout << "1 - вывести список и кол-во элементов;" << endl;  cout << "2 - найти элемент;" << endl;  cout << "3 - удалить элемент;" << endl;  cout << "4 - удаление списка;" << endl;  cout << "0 - выход." << endl;  cin >> c;  switch (c)  {  case 1:  L1.PrintList(print);  break;  case 2: {  cout << "Сделайте выбор:" << endl;  cout << "1 элемент:" << endl;  cout << "2 элемент:" << endl;  cout << "3 элемент:" << endl;  cin >> n;  switch (n) {  case 1: {  Element\* e = L1.Search(&a1);  HotelClients\* aa = (HotelClients\*)e->Data;  cout << "ФИО №1: " << aa->fio << endl; }  break;  case 2: {  Element\* e = L1.Search(&a2);  HotelClients\* aa = (HotelClients\*)e->Data;  cout << "ФИО №2: " << aa->fio << endl; }  break;  case 3: {  Element\* e = L1.Search(&a3);  HotelClients\* aa = (HotelClients\*)e->Data;  cout << "ФИО №3: " << aa->fio << endl; }break;  }break;  }  case 3: {  cout << "Сделайте выбор:" << endl;  cout << "1 элемент:" << endl;  cout << "2 элемент:" << endl;  cout << "3 элемент:" << endl;  int o;  cin >> o;  switch (o) {  case 1: rc = L1.Delete(&a1);  if (rc == true) cout << "Удален элемент №1." << endl;  break;  case 2:rc = L1.Delete(&a2);  if (rc == true) cout << "Удален элемент №2." << endl;  break;  case 3: rc = L1.Delete(&a3);  if (rc == true) cout << "Удален элемент №3." << endl;  break;  }break;  cout << "Список:" << endl;  L1.PrintList(print);  return 0;  }  case 4:  rc = L1.Delete(&a1);  rc = L1.Delete(&a2);  rc = L1.Delete(&a3);  break;  }  } while (c != 0);  } |
| Файл с функциями |
| #include "List.h"  bool Object::Insert(void\* data)  {  bool rc = 0;  if (Head == NULL)  {  Head = new Element(NULL, data, Head);  rc = true;  }  else  {  Head = (Head->Prev = new Element(NULL, data, Head));  rc = true;  }  return rc;  }  //-----------------------------------------------------------  Element\* Object::Search(void\* data)  {  Element\* rc = Head;  while ((rc != NULL) && (rc->Data != data))  rc = rc->Next;  return rc;  }  //-----------------------------------------------------------  void Object::PrintList(void(\*f)(void\*))  {  int h = 0;  Element\* e = Head;  while (e != NULL)  {  f(e->Data);  e = e->GetNext();  h = h + 1;  };  cout << "Количество элементов равно: " << h << ".\n";  }  //-----------------------------------------------------------  void Object::PrintList(void(\*f)(void\*), Element\* e)  {  f(e->Data);  }  //-----------------------------------------------------------  bool Object::Delete(Element\* e)  {  bool rc = 0;  if (rc = (e != NULL))  {  if (e->Next != NULL)  e->Next->Prev = e->Prev;  if (e->Prev != NULL)  e->Prev->Next = e->Next;  else  Head = e->Next;  delete e;  }  return rc;  }  //-----------------------------------------------------------  bool Object::Delete(void\* data)  {  return Delete(Search(data));  };  //-----------------------------------------------------------  Element\* Object::GetLast()  {  Element\* e = Head, \* rc = e;  while (e != NULL)  {  rc = e;  e = e->GetNext();  }  return rc;  }  //-----------------------------------------------------------  Object Create()  {  return \*(new Object());  }  //----------------------------------------------------------- |
| Файл заголовков List.h |
| #pragma once  #include <iostream>  using namespace std;  struct Element  {  Element\* Prev;  Element\* Next;  void\* Data;  Element(Element\* prev, void\* data, Element\* next)  {  Prev = prev;  Data = data;  Next = next;  }  Element\* GetNext()  {  return Next;  };  Element\* GetPrev()  {  return Prev;  };  };  //-----------------------------------------------------------  struct Object  {  Element\* Head;  Object()  {  Head = NULL;  };  Element\* GetFirst()  {  return Head;  };  Element\* GetLast();  Element\* Search(void\* data);  bool Insert(void\* data);  bool Delete(Element\* e);  bool Delete(void\* data);  void PrintList(void(\*f)(void\*));  void PrintList(void(\*f)(void\*), Element\*);  };  Object Create(); |
| Консоль отладки |
|  |

2. Каждый элемент списка студентов содержит фамилию, имя, отчество, год рождения, курс, номер группы, оценки по пяти предметам. Упорядочить студентов по курсу, причем студенты одного курса должны располагаться в алфавитном порядке. Найти средний балл каждой группы по каждому предмету. Определить самого старшего студента и самого младшего студентов. Для каждой группы найти лучшего с точки зрения успеваемости студента.

|  |
| --- |
| Программный код |
| #include <iostream>  #include <algorithm>  #include <windows.h>  using namespace std;  struct item  {  int value;  char fname[30] = {};  char sname[30] = {};  char thname[30] = {};  int year;  int course;  int group;  int marks[5];  float middle\_mark = 0;  item\* next;  item\* prev;  };  item\* plist = NULL;  item\* p;  item\* plist1 = NULL;  item\* p1;  int count1 = 0;//подсчет всех элементов  int counter = 1;//для функции вывода  void add()//добавить элемент(студента)  {  cout << "======================================" << endl;  char fname[50];  char sname[50];  char thname[50];  cout << "Введите ФИО студента: ";  cin >> fname >> sname >> thname;  int year;  cout << "Введите год рождения: ";  cin >> year;  int group, course, marks[5];  cout << "Введите курс и номер группы: ";  cin >> course >> group;  for (int i = 0; i < 5; i++)  {  cout << "Введите оценку номер №" << i + 1 << ": ";  cin >> marks[i];  }  p = new item;  for (int i = 0; i < strlen(fname); i++)  {  p->fname[i] = fname[i];  }  for (int i = 0; i < strlen(sname); i++)  {  p->sname[i] = sname[i];  }  for (int i = 0; i < strlen(thname); i++)  {  p->thname[i] = thname[i];  }  p->year = year;  p->course = course;  p->group = group;  for (int i = 0; i < 5; i++)  {  p->marks[i] = marks[i];  p->middle\_mark += marks[i];  }  p->middle\_mark /= 5;  p->next = plist;  p->prev = NULL;  if (p->next != NULL)  {  item\* previ;  previ = p;  p = p->next;  p->prev = previ;  p = previ;  }  plist = p;  count1++;  cout << "Студент был успешно добавлен в начало списка." << endl;  cout << "======================================" << endl;  return;  }  void Sort()  {  cout << "======================================" << endl;  int val = count1 - 1;  for (int i = 0; i < count1; i++)  {  item\* host1 = p;  item\* host2 = p->next;  for (int j = 0; j < val; j++)  {  if (host1->course < host2->course)  {  swap(host1->fname, host2->fname);  swap(host1->sname, host2->sname);  swap(host1->thname, host2->thname);  swap(host1->course, host2->course);  swap(host1->group, host2->group);  swap(host1->year, host2->year);  swap(host1->marks, host2->marks);  swap(host1->middle\_mark, host2->middle\_mark);  }  host1 = host1->next;  host2 = host2->next;  }  val--;  }  cout << "Сортировка проведена успешно!" << endl;  cout << "======================================" << endl;  }  void write()  {  cout << "======================================" << endl;  item\* host = p;  for (int i = 0; i < count1; i++)  {  cout << "ФИО: " << host->fname << " " << host->sname << " " << host->thname << endl;  cout << "Год рождения: " << host->year << " Курс: " << host->course << " Группа: " << host->group << endl;  cout << "Оценки: " << host->marks[0] << " " << host->marks[1] << " " << host->marks[2] << " " << host->marks[3] << " " << host->marks[4] << endl;  host = host->next;  }  cout << "======================================" << endl;  }  void better\_student()  {  cout << "======================================" << endl;  item\* host = p;  int max\_mark = host->middle\_mark;  item\* better = host;  for (int i = 0; i < count1; i++)  {  if (max\_mark < host->middle\_mark)  {  max\_mark = host->middle\_mark;  better = host;  }  host = host->next;  }  cout << "ФИО лучшего по успеваемости студента: " << better->fname << " " << better->sname << " " << better->thname << " " << max\_mark << endl;  cout << "======================================" << endl;  }  void max\_min\_year()  {  cout << "======================================" << endl;  item\* host = p;  int ma = host->year;  int mi = host->year;  item\* mi\_st = host;  item\* ma\_st = host;  for (int i = 0; i < count1; i++)  {  if (ma < host->year)  {  ma = host->year;  ma\_st = host;  }  if (mi > host->year)  {  mi = host->year;  mi\_st = host;  }  host = host->next;  }  cout << "Самый старший студент: " << mi\_st->fname << " " << mi\_st->sname << " " << mi\_st->thname << endl;  cout << "Самый младший студент: " << ma\_st->fname << " " << ma\_st->sname << " " << ma\_st->thname << endl;  cout << "======================================" << endl;  }  void middle\_groups()  {  cout << "======================================" << endl;  item\* ho = p;  for (int i = 0; i < count1; i++)  {  p1 = new item;  for (int i = 0; i < strlen(ho->fname); i++)  {  p1->fname[i] = ho->fname[i];  }  for (int i = 0; i < strlen(ho->sname); i++)  {  p1->sname[i] = ho->sname[i];  }  for (int i = 0; i < strlen(ho->thname); i++)  {  p1->thname[i] = ho->thname[i];  }  p1->year = ho->year;  p1->course = ho->course;  p1->group = ho->group;  for (int i = 0; i < 5; i++)  {  p1->marks[i] = ho->marks[i];  }  p1->middle\_mark = ho->middle\_mark;  p1->next = plist1;  p1->prev = NULL;  if (p1->next != NULL)  {  item\* previ;  previ = p1;  p1 = p1->next;  p1->prev = previ;  p1 = previ;  }  plist1 = p1;  ho = ho->next;  }  int val = count1 - 1;  for (int i = 0; i < count1; i++)  {  item\* host11 = p1;  item\* host22 = p1->next;  for (int j = 0; j < val; j++)  {  if (host11->group < host22->group)  {  swap(host11->fname, host22->fname);  swap(host11->sname, host22->sname);  swap(host11->thname, host22->thname);  swap(host11->course, host22->course);  swap(host11->group, host22->group);  swap(host11->year, host22->year);  swap(host11->marks, host22->marks);  swap(host11->middle\_mark, host22->middle\_mark);  }  host11 = host11->next;  host22 = host22->next;  }  val--;  }  item\* host = p1;  item\* best = host;  int ma = host->middle\_mark, gr = host->group;  for (int i = 0; i < count1; i++)  {  if (host->group != gr)  {  cout << "Студент группы №" << gr << " лучший по успеваемости: " << best->fname << " " << best->sname << " " << best->thname << endl;  gr = host->group;  ma = host->middle\_mark;  best = host;  }  if (ma < host->middle\_mark)  {  ma = host->middle\_mark;  gr = host->group;  best = host;  }  if (i != count1 - 1)host = host->next;  }  if (ma < host->middle\_mark)  {  ma = host->middle\_mark;  gr = host->group;  best = host;  }  cout << "Студент группы №" << gr << " лучший по успеваемости: " << best->fname << " " << best->sname << " " << best->thname << endl;  cout << "======================================" << endl;  }  void main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_CTYPE, "Rus");  int c;  do {  cout << "Что собираетесь делать?" << endl;  cout << "1 - Добавить студента;" << endl;  cout << "2 - Вывести всех студентов;" << endl;  cout << "3 - Сортировка в порядке курсов;" << endl;  cout << "4 - Лучший студент по успеваемости;" << endl;  cout << "5 - Самый старший и младший студент;" << endl;  cout << "6 - Средний балл по группам;" << endl;  cout << "7 - Выход." << endl;  cin >> c;  switch (c)  {  case 1:  add();  break;  case 2:  write();  break;  case 3:  Sort();  break;  case 4:  better\_student();  break;  case 5:  max\_min\_year();  break;  case 6:  middle\_groups();  break;  case 7:  cout << "До свидания!" << endl;  break;  default:  cout << "Некорректный запрос, повторите пожалуйста." << endl;  break;  }  } while (c != 7);  } |
| Консоль отладки |
|  |

3. N человек  располагаются  по кругу.  Начав отсчет от первого, удаляют каждого k-го, смыкая круг после удаления. Определить порядок удаления людей из круга. Использовать линейный список.

|  |
| --- |
| Программный код |
| #include <iostream>  using namespace std;  // объявим струткуру  struct numbers  {  char num[10]; // для циферок  numbers\* next; // для следующего  };  char\* circle(numbers\*, int, int);  numbers\* zapoln(int);  int main()  {  setlocale(LC\_ALL, "ru");  int kolv;  int k;  cout << "Введите количество участников: ";  cin >> kolv;  numbers\* g = zapoln(kolv); // заполним очередь участников  cout << "Введите k: ";  cin >> k;  cout << "Последним остался: " << circle(g, k, kolv) << endl; // из цикла выведем последнего оставшегося  return 0;  }  numbers\* zapoln(int kolv)  {  numbers\* first = nullptr;  numbers\* Head;  numbers\* nowiy;  char a[10];  if (kolv > 0) // в случае если > 0  {  nowiy = new numbers;  Head = nowiy;  }  else  {  return first;  }  for (int i = 0; i < kolv; i++) // цикл для определения порядкового номера  {  cout << "Введите порядковый номер: ";  cin >> a;  for (int y = 0; y < 10; y++)  {  if (a[y] == '\0')  {  nowiy->num[y] = '\0';  break;  }  nowiy->num[y] = a[y]; // присваиваем значение  }  first = nowiy;  nowiy = new numbers;  first->next = nowiy; // перемещаемся  }  first->next = Head;  return Head;  }  char\* circle(numbers\* g, int ocher, int kolv)  {  int u = 0;  numbers\* buff;  numbers\* h = g;  while (h->next != h) // пока не пройдем весь круг  {  for (int i = 0; i < ocher - 2; i++)  {  h = h->next; // с очередью на 2 элемента будем сдвигаться  }  buff = h->next;  cout << "Удалён " << buff->num << endl;  h->next = h->next->next;  h = h->next;  delete buff;  }  return h->num;  } |
| Консоль отладки |
|  |