ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра программной инженерии

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ

«Объектно-ориентированное программирование»

(направление подготовки 6.09.03.04 «Программная инженерия»)

Донецк, 2016

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

 «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра программной инженерии

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ

«Объектно-ориентированное программирование»

(направление подготовки 6.09.03.04 «Программная инженерия»)

Рассмотрено на заседании кафедры

Программной инженерии

Протокол № \_1\_\_ от «\_30\_\_»\_\_08\_\_\_2016 г.

Донецк, 2016

# УДК 681.142.2

Методические указания и задания к выполнению лабораторных работ по курсу «Объектно-ориентированное программирование» (направление подготовки 6.09.03.04 «Программная инженерия»)/ Сотовит: И.А. Коломойцева, О.В. Морозова - Донецк, ДонНТУ, 2016. - 37 с.

 Данные задания и указания к их выполнению к таким темам дисциплины “ Объектно-ориентированное программирование”: объявление класса при помощи C++, имитационное моделирование, простое наследование и полиморфизм, перегрузка операций, шаблоны классов. Для каждой темы приведен пример и его реализация на ЭВМ и программа на языке С++.

Составитель И.А. Коломойцева ст. пр.

 О.В. Морозова асс.

Рецензент В.И. Грищенко, доц.

Содержание

Лабораторная работа №1 Описание класса на языке C++

Лабораторная работа № 2 Имитационное моделирование средствами языка С++

Лабораторная работа №3 Простое наследование и полиморфизм

# Лабораторная работа №4 Перегрузка операций

# Лабораторная работа №5 Шаблоны классов

Лабораторная работа №1

**Описание класса на языке C++**

**Цель работы:** изучить структуру классов, особенности реализации методов класса и создания объектов; приобрести навыки в работе с классами на языке C++.

**Задание.**

Согласно заданию программно объявить класс на языке C++, его методы и поля, написать реализацию этих методов.

В каждом классе должен быть:

- конструктор (задаёт начальные значения полям класса);

- деструктор (освобождает память, которая выделялась под массив);

- метод, добавляющий элемент в список;

- метод, очищающий список;

- метод, выводящий список на экран;

- метод, описанный в варианте заданий.

В функции main() создать объект этого класса, с его помощью продемонстрировать работу всех методов класса. Вызов каждого метода (кроме конструктора и деструктора) организовать с помощью меню.

При реализации методов класса предусмотреть обработку ошибочных ситуаций: в списке нет элемента с номером N, N меньше 0 и т.п.

**Пример определения класса и работы с его объектом**

//файл list.h

**// объявление класса**

class CList

{

 struct list

 {

 char info[100];

 struct list \*next;

 };

 struct list \*head;

public:

 CList();

 ~CList();

 void AddItem(char \*);

 void DisplayList();

 void ReleaseList();

};

//файл list.cpp

#include "array.h"

#include <stdlib.h>

#include <stdio.h>

**//реализация методов класса**

#include <string.h>

#include <stdio.h>

#include "list.h"

CList::CList()

{

 head=NULL;

}

CList::~CList()

{

 ReleaseList();

}

void CList::AddItem(char\* s)

{

 struct list \*newItem;

 newItem = new struct list;

 strcpy(newItem->info,s);

 if (head==NULL)

 {

 head=newItem;

 head->next = NULL;

 }

 else

 {

 newItem->next = head;

 head = newItem;

 }

}

void CList::DisplayList()

{

 char \*s=new char[100];;

 char s1[10];

 struct list \*p;

 p=head;

 strcpy(s,"");

 while (p!=NULL)

 {

 printf("%s->",p->info);

 p=p->next;

 }

 printf("NULL\n");

}

void CList::ReleaseList()

{

 if (head!=NULL)

 {

 struct list \*p;

 while (head!=NULL)

 {

 p=head;

 head=head->next;

 delete p;

 }

 head=NULL;

 }

}

//файл mainfile.cpp

#include "array.h"

#include <stdio.h>

#include <conio.h>

#include "list.h"

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <stdlib.h>

int main()

{

 CList list; //создаётся объект класса, при этом автоматически вызывается конструктор

 int key;

 char val[100];

 setlocale(0,"RUS"); // обеспесиваем вывод в консольное окно символов кириллицы

 do

 {

 printf("\n\n1 - добавление элемента\n");

 printf("2 - вывод списка на экран\n");

 printf("3 - очитска списка\n");

 printf("ESC - выход\n");

 printf("Ваш выбор: \n");

 key=getch();

 switch(key)

 {

 case '1':

 printf("Введите элемент списка: ");

 scanf("%s",val);

 list.AddItem(val);

 break;

 case '2':

 list.DisplayList();

 break;

 case '3':

 list.ReleaseList();

 break;

 }

 }

 while(key!=27);

 return 0; //объект класса уничтожается, автоматически вызывается деструктор

}

**Содержание отчёта**

1. Титульный лист.
2. Условие лабораторной работы.
3. Текст программы.
4. Экранные формы с примерами работы программы.

**Варианты заданий.**

1. Описать класс CListDelete для работы со списками. Дополнительный метод: удаление N-го элемента списка (N – аргумент метода, задаётся пользователем в функции main()).

2. Описать класс CListLast для работы со списками. Дополнительный метод: вывод последних N элементов списка.

3. Описать класс CListDoubleDirected для работы с двунаправленными списками. Дополнительный метод: вывод элементов списка на экран обратном направлении.

4. Описать класс CListRemoveIfOdd для работы со списками. Дополнительный метод: удаление элементов, стоящих на четных местах.

5. Описать класс CListRemoveIfOddValue для работы со списками. Дополнительный метод: удаление элементов, четных по значению.

6. Описать класс CListRemoveIfNotOdd для работы со списками. Дополнительный метод: удаление элементов, стоящих на нечетных местах.

7. Описать класс CListRemoveIfOddValue для работы со списками. Дополнительный метод: удаление элементов, нечетных по значению.

8. Описать класс CListRemoveEveryN для работы со списками. Дополнительный метод: удаление каждого N-го элемента (N – аргумент метода, задаётся пользователем в функции main()).

9. Описать класс CListSorted для работы с упорядоченными списками. Дополнительный метод: добавление элементов в список с сохранением упорядоченности списка.

10. Описать класс CListDeleteFirstZero для работы со списками. Дополнительный метод: удаление из массива первого найденного нулевого элемента со сдвигом элементов.

11. Описать класс CListShiftLeft для работы со списками. Дополнительный метод: циклический сдвиг элементов списка влево на N позиций (N – аргумент метода, задаётся пользователем в функции main()).

12. Описать класс CListShiftRight для работы со списками. Дополнительный метод: циклический сдвиг элементов списка вправо на N позиций (N – аргумент метода, задаётся пользователем в функции main()).

13. Описать класс CListDeleteMaxPositiveItems для работы со списками. Дополнительный метод: удаление из списка максимального положительного элемента.

14. Описать класс CListDeleteMaxNegativeItems для работы со списками. Дополнительный метод: удаление из списка максимального отрицательного элемента.

15. Описать класс CListDeleteMinPositiveItems для работы со списками. Дополнительный метод: удаление из списка минимального положительного элемента.

16. Описать класс CListDeleteMinNegativeItems для работы со списками. Дополнительный метод: удаление из списка минимального отрицательного элемента.

17. Описать класс CListDeleteMaxOddItems для работы со списками. Дополнительный метод: удаление из списка максимального чётного элемента.

18. Описать класс CListDeleteMaxNotOddItems для работы со списками. Дополнительный метод: удаление из списка максимального нечётного элемента.

19. Описать класс CListDeleteMinOddItems для работы со списками. Дополнительный метод: удаление из списка минимального чётного элемента.

20. Описать класс CListDeleteMinNotOddItems для работы со списками. Дополнительный метод: удаление из списка минимального нечётного элемента.

21. Описать класс CListDeleteFirstOdd для работы со списками. Дополнительный метод: удаление из списка первого найденного чётного элемента.

22. Описать класс CListDeleteFirstNotOdd для работы со списками. Дополнительный метод: удаление из массива первого найденного нечётного элемента.

23. Описать класс CListDeleteFirstPositive для работы со списками. Дополнительный метод: удаление из списка первого найденного положительного элемента.

24. Описать класс CListDeleteFirstNegative для работы со списками. Дополнительный метод: удаление из списка первого найденного отрицательного элемента со сдвигом элементов.

25. Описать класс CListDeleteLastOdd для работы со списками. Дополнительный метод: удаление из списка последнего найденного чётного элемента.

Лабораторная работа № 2

Имитационное моделирование средствами языка С++

Цель работы: приобрести навыки в определении класса, инкапсулирующего объект из реального мира.

Задание. Разработать класс на языке C++, моделирующий объект, описанный в варианте заданий. Этот класс обязательно должен содержать метод, который обеспечивает управление объектом со стороны пользователя. Вызов только этого метода для объекта разработанного класса должен присутствовать в функции main(). Любое изменение состояния объекта, описанного в задании, должно отображаться на экране.

**Пример. Работа с микроволновой печью.**

Описание системы:

1. имеется единственная кнопка управления, которая доступна для пользователя печи. Если дверь печи закрыта, и пользователь нажимает кнопку, то печь будет готовить пищу в течение 1 минуты;
2. если пользователь нажимает на кнопку во время работы печи, получаем дополнительную минуту работы;
3. если дверь открыта, нажатие кнопки не имеет эффекта ;
4. внутри печи есть электролампа, во время работы она должна быть включена; когда дверь печи открыта, электролампа должна быть включена;
5. можно приостановить процесс приготовления пищи открытием двери, в этом случае время сбрасывается;
6. начальная конфигурация: дверь закрыта, лампа погашена;
7. если время истекло, выключается питание и электролампа.

Функцию управления берет на себя пользователь, то есть управление осуществляется с клавиатуры.

Идентификаторы состояния:

1. включена/выключена электролампа;
2. включено/выключено электропитание;
3. открыта/закрыта дверь;
4. сколько времени осталось до конца приготовления пищи.

|  |  |  |
| --- | --- | --- |
| Электролампа | Дверь | Питание |
| 0 | 0 | Невозможный набор |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Необходимые функции (сообщения):

1. нажать кнопку;
2. изменить положение двери;
3. пища готова.

**//файл stove.h**

enum state {on, off}; //определение типа для элемента, поддерживающего состояние включён/выключен

enum door {open, close}; //определение типа для элемента, поддерживающего состояние открыт/закрыт

class stove

{ state curr\_lamp; //поле для описания состояния лампы

 door curr\_door; //поле для описания состояния двери

 state curr\_power; //поле для описания состояния питания

 int time; //время, оставшееся до конца приготовления еды

 void take\_door(void); //метод, изменяющий положение двери

 void push\_button(void); //метод, обрабатывающий работы с питанием микроволновой печи

 void ready(void); //метод, обрабатывающий окончание приготовления еды

public:

 stove(); //конструктор, задёт начальные параметры

 void Run(); //метод, обрабатывающий взаимодействие пользователя с печью

};

**//файл stove.cpp**

#include <stdio.h>

#include <conio.h>

#include <windows.h>

#include "stove.h"

void menu() //выводит на экран меню

{

 printf("\n%b - нажать кнопку для приготовления еды\n");

 printf("d - открыть/закрыть дверь\n");

 printf("ESC - выход\n\n");

}

stove::stove()

{

//начальноге состояние:

 curr\_lamp=off; //лампа выключена

 curr\_door=close; //дверь закрыта

 curr\_power=off; //питание выключено

 time=-1; //еда не готовится

}

void stove::take\_door(void)

{

 if (curr\_door==open) //если дверь открыта

 { curr\_door=close; //закрываем дверь

 curr\_lamp=off; //выключаем лампу

 printf("Дверь закрыта\n");

 }

 else //если дверь закрыта

 { curr\_door=open; //открываем дверь

 curr\_lamp=on; //включаем лампу

 printf("Дверь открыта\n");

 if (curr\_power==on) //если в момент открытия двери готовилась еда

 {

 printf("Процесс прерван\n");

 time=-1; //сбрасываем счётчик времени

 curr\_power=off; //выключаем питание

 }

 }

 return;

}

void stove::push\_button(void)

{

 if (curr\_door==open) //если дверь открыта, еду готовить нельзя

 printf("Закройте дверь!\n");

 else //если дверь закрыта

 { if (curr\_power==on) //если еда уже готовится

 { time+=60; //добавляем минуту

 printf("Вам добавлена минута\n");

 }

 else //если еда не готовится

 { curr\_power=on; //включаем питание

 curr\_lamp=on; //включаем лампу

 time=60; //устанавливаем счётчик времени

 printf("Еда готовится\n");

 }

 }

 return;

}

void stove::ready(void)

{

//еда пригтовлена

 curr\_power=off; //выключаем питание

 curr\_lamp=off; //выключаем лампу

 time=-1; //сбрасываем счётчик времени

 printf("\aЕда готова\n");

}

void stove::Run(void)

{

 int key;

 do

 {

 while (!kbhit() && time>0) //в этот цикл заходим, если еда готовится;

условие выхода из цикла: нажата клавиша или истекло время приготовления еды

//функция kbhit() возвращает истину, если пользователь нажал клавишу

 {

 time--; //вычитаем секунду из времени, оставшегося до конца приготовления еды

 Sleep(100); //задержка в 100 мс

 printf("Осталось %d секунд\n",time);

 }

 if (time==0) //если еда закончила готовиться, вызываем метод ready()

 ready();

 menu();

 key=getch();

 switch (key)

 { case 'b': push\_button(); break;

 case 'd': take\_door(); break;

 }

 fflush(stdin);

 }

 while(key!=27);

 return;

}

**//файл mainfile.cpp**

#include "stove.h"

#include <conio.h>

#include <locale.h>

int main()

{

 setlocale(0,"RUS");

 stove MyStove;

 MyStove.Run();

 return 0;

}

**Содержание отчёта**

1. Титульный лист.
2. Условие лабораторной работы.
3. Текст программы.
4. Экранные формы с примерами работы программы.

**Варианты заданий.**

1. Трамвай

Имеется две кнопки: управления: газ и открытие/закрытие дверей. Если двери трамвая закрыты и водитель нажимает «газ», то трамвай начинает движение в пределах одной минуты. Если водитель во время движения нажимает «газ», то трамвай дополнительно движется одну минуту. Если двери открыты, то трамвай с места тронутся не может. Внутри трамвая есть освещение, которое работает во время остановки и отключается во время движения (но не во время простоя с закрытыми дверями). Во время движения двери открыться не могут. Если время прошло трамвай останавливается. Начальная конфигурация: трамвай стоит, двери закрыты.

2. Лифт

В лифте есть 9 кнопок, соответствующих этажам, и кнопка открытия/закрытия дверей. Нажатием на каждую кнопку сопровождается движение лифта на соответствующий этаж. Движение от этажа к этажу осуществляется в течение одной минуты (в условных единицах времени). Двери могут открываться во время остановки и не могут – во время движения. При нажатии кнопки этажа с открытыми дверьми двери автоматически закрываются. С одного этажа на один и тот же этаж лифт в движение не приводится. Счётчик этажей реализовать визуально. Начальная конфигурация: лифт на первом этаже двери открыты.

3. Светофор

Светофор может иметь 4 состояния: «красный», «жёлтый», «зелёный», «жёлтый мигающий». Светофор управляется вручную оператором с помощью 3-х кнопок. Две кнопки – это кнопки управления движением ⇑ и ⇓, которыми оператор переключает состояния «красный», «жёлтый», «зелёный» в соответствии с их нормальной последовательностью (кр-жёл-зел или зел-жёл-кр, кр-жёл-кр, зел-жёл-зел, но не кр-зел, зел-кр). При нажатии на третью кнопку «работа» светофор переключается с любого состояния в состояние «жёлтый мигающий». В этом состоянии нажатие на кнопки ⇑ и ⇓ ничего не вызывает. Из состояния «жёлтый мигающий» нажатием кнопки «работа» светофор переключается в состояние «жёлтый». Начальная конфигурация состояние: «жёлтый мигающий».

4. Принтер

Принтер имеет две кнопки: «печать» и «заправка бумаги». Печать происходит в течение 1 минуты. Печать без заправленной бумаги производиться не может. Во время печати можно заправить ещё один лист. Во время печати нажатием на кнопку «печать» ничего не производится. Если бумага заправлена, то повторно заправляться не может. Начальная конфигурация: принтер без бумаги.

5. Автомат Калашникова.

Автомат имеет спусковой механизм (курок) и рычаг переключения «одиночный выстрел» ‑ «автоматическая стрельба». При нажатии на курок в зависимости от рычага переключения автомат выстреливает:

* 1 пулю при «одиночный выстрел»;
* 5 пуль при «автоматическая стрельба».

Если обойма пуста, то стрельба производится не может и необходимо сменить обойму. Ёмкость обоймы 30 патронов. Если в обойме осталось менее 5 патронов, то при «автоматической стрельбе» выстреливаются все остающиеся. Две обоймы заправлять одновременно нельзя. Начальное состояние: обойма полна т.е. 30 патронов и рычаг переключен в состояние «одиночный выстрел».

6. Кондиционер

Кондиционер имеет 3 кнопки: кнопка режима (холодный/горячий воздух); кнопка питания; кнопка работы. Кнопка режима действует только при выключенном питании. Кнопка работы приводит кондиционер в работу на одну минуту и только когда включено питание. Повторное нажатие на эту кнопку добавляет ещё одну минуту времени. При включенном питании нажатие на кнопку питания приводит к отключению кондиционера, обнулению счётчика времени и выключению питания. При выключенном питании нажатие кнопки питания приводит к включению кондиционера. Начальное состояние: питание выключено, режим – холодный воздух.

7. Стиральная машина

Стиральная машина имеет три кнопки: кнопка «стирка», кнопка «отжим», кнопка «режим». Кнопка «режим» предназначена для стирки и переключает режимы: «нормальная стирка» и «споласкивание». При нажатии на кнопку «стирка» машина начинает стирать (споласкивать) в зависимости от режима: 5 минут, если «нормальная стирка», и 3 минуты, если «споласкивание». Во время стирки нажатие на любые кнопки ни к чему не приводит. При нажатии на кнопку «отжим» машина начинает выжимать бельё в течение 3 минут. Во время отжима можно переключать режим стирки. Во время простоя можно нажимать все три кнопки. Начальная конфигурация: режим «нормальная стирка».

8. Фотоаппарат

Фотоаппарат имеет две кнопки: «фото» и «перемотка». Фотоплёнка имеет ёмкость 10 кадров. Нажатием на кнопку «фото» фотоаппарат делает снимок. При нажатии на кнопку «перемотка» фотоаппарат прокручивает плёнку на один кадр вперёд. Без перемотки делать снимок фотоаппарат не может (кроме первого кадра). Без снимка фотоаппарат также не может сделать перемотку. После снимка последнего, 10 кадра, перемотка не работает, пока не будет заменена фотоплёнка. Пока не будет отснята вся плёнка, её также нельзя заменить. Если фотоаппарат пуст (нет плёнки), то нажатие на кнопки ничего не дает. Начальная конфигурация: фотоаппарат пуст.

9. Троллейбус

Имеется три кнопки управления: «газ», открытие/закрытие дверей и «свет в салоне». Если двери троллейбуса закрыты и водитель нажимает «газ», то он начинает движение в пределах одной минуты. Если водитель во время движения нажимает «газ», то троллейбус дополнительно движется одну минуту. Если двери открыты, то троллейбус с места тронутся не может. Во время движения двери открываться не могут. Если время истекло, троллейбус останавливается. Двери можно открыть, только включив свет в салоне, закрыть, только когда свет отключен. Начальная конфигурация: троллейбус стоит, двери закрыты, освещение выключено.

10. Спортивный тренажёр

Тренажёр представляет собой снаряд для бега или ходьбы со звуковым сопровождением шагов. Имеется 2 кнопки управления: кнопка «время +1 мин», «режим бег/ходьба» и переключатель скорости 1-2-3. Кнопка «время +1 мин» включает тактовый генератор на 1 минуту или добавляет 1 мин, если он был уже включен. В режиме шагов генератор выдаёт импульсы с частотой 60 шагов в минуту не зависимо от показания тумблера скорости. В режиме «бег» генератор создаёт импульсы с частотой:

* 120 шагов в минуту на 1 скорости;
* 180 шагов в минуту на 2 скорости;
* 240 шагов в минуту на 3 скорости.

При этом во время бега тумблер скорости можно переключать только после 1 минуты бега. Начальная конфигурация: «ходьба» , 1-я скорость.

11. Переносная печь «Мечта»

Печь имеет левую и правую конфорки и духовку. Одновременно могут работать только:

* обе конфорки;
* левая конфорка и духовка;
* правая конфорка и духовка.

При этом имеется четыре кнопки управления:

* кнопка «+1 минута работы левой конфорки»;
* кнопка «+1 минута работы правой конфорки»;
* кнопка «+1 минута работы духовки»;
* кнопка питания.

Первые 3 кнопки включают (или добавляют по одной минуте работы для соответствующей конфорки (духовки) при условии, что кнопка питания включена). Если во время работы печи была выключена кнопка питания, то печь перестаёт работать, все три счётчика обнуляются. Начальная конфигурация: печь выключена.

12. Электричка

Электричка движется по железной дороге в левом и правом направлениях минуя 6 станций, 1-я и 6-я из которых являются конечными. Время движения от одной станции до другой составляет 1 минуту. Во время движения машинист не может открыть двери. На пульте управления электричкой имеются кнопки «газ», «открытие/закрытие дверей» и «правое/левое направление». На конечных станциях машинист должен изменить направление движения на противоположное после остановки поезда. В средине маршрута изменить направление следования нельзя. Достигнув остановки машинист может открыть двери или по усмотрению двигаться к следующей станции, причём на конечных станциях машинист обязан выпустить пассажиров. Поезд может тронуться только с закрытыми дверями нажатием кнопки «газ». Во время движения нажатие на кнопки не даёт результата. Станциям желательно дать название. Начальная конфигурация: поезд на 1-й станции, правое направление движения, двери закрыты, салон пуст.

13. СВД (снайперская винтовка Драгунова)

Винтовка имеет спусковой крючок (курок), подстраиваемый оптический прицел 100–1000 м. При нажатии курка производится выстрел, причём время следования пули рассчитывается по формуле: (мкс), где x – расстояние до цели (показания оптического прицела). После выстрела необходимо вывести на экран время следования пули. После того как обойма закончится, её необходимо заменить. Объём обоймы – 10 патронов. Начальная конфигурация: полная обойма (10 патронов), прицел – 100 м.

14. Холодильник

Холодильник имеет переключатель режима холода 1–2–3. Внутри холодильника имеется освещение, которое включается при открытии дверцы и отключается при её закрытии. Холодильник работает по принципу: X мин компрессор работает (Х – режим холода), 1 минуту не работает (простой) и так далее по циклу. Переключатель расположен внутри холодильника, таким образом, чтобы переключить режим, необходимо открыть дверцу, при этом, после изменения режима работы счётчик времени обнуляется и запускается в новом режиме. Начальная конфигурация: холодильник закрыт, режим работы 1.

15. Автобус

Имеется три кнопки управления: «газ», «тормоз» и открытие/закрытие дверей. Если двери автобуса закрыты и водитель нажимает «газ», то автобус начинает движение в пределах одной минуты. Если водитель во время движения нажимает «газ», то автобус дополнительно движется одну минуту. Если во время движения нажать клавишу тормоз, то автобус останавливается, при этом обнуляется счётчик движения. Если двери открыты, то автобус с места тронутся не может. Внутри есть освещение, которое горит во время остановки и тухнет во время движения (но не во время простоя с закрытыми дверями). Во время движения двери открыться не могут. Если время истекло, автобус останавливается. Начальная конфигурация: автобус стоит, двери закрыты.

16. Телефон

Телефон имеет набор цифр от 0 до 9. Номер набирается только при поднятой трубке. При наборе номера проверяется последовательность и количество набранных цифр в зависимости от первой. Возможные варианты:

* 01, 02, 03, 04;
* 1 – сигнал «короткие гудки»;
* 2, 4, 5, 6, 7, 9 – количество цифр в наборе – шесть;
* 3 – количество цифр в наборе – семь;
* 8 – после набора должен быть непрерывный гудок, и потом 10 цифр.

Состояние трубки – «нет гудка», «непрерывный гудок», «длинные гудки», «короткие гудки», должно отображаться на экране после каждой набранной цифры. Чтобы перезвонить, необходимо положить и снова снять трубку. Начальная конфигурация: трубка лежит на телефоне.

17. Автоматика снабжения котельной

Пульт управления автоматикой котельной содержит 4 кнопки управления: «вкл/выкл горелки», «+25 градусов», «–25 градусов» и «пуск».

Кнопки «+25 градусов» и «–25 градусов» увеличивают или уменьшают температуру горения котла в интервалах 100–300°C для нормальной работы котла. Причём, если температура ниже 150°C или выше 250°C, загорается красная лампочка. Если температура находится в интервале 150–250°C, то лампочка не горит. При температуре свыше 300°C котёл взрывается, а при температуре менее 100°C останавливается. Котел может работать только со включёнными горелками. Если во время работы выключить горелки, котёл останавливается, при этом счётчик температуры не меняется. Запускается котёл при температуре 100 – 150°C при включённых горелках с помощью кнопки «пуск», в остальных случаях кнопка «пуск» не работает. Начальная конфигурация: котёл не работает, горелки отключены, выставлена температура 100°C, горит красная лампочка.

18. Игра «крестики-нолики»

Игровое поле состоит из столбцов А, B, C и строк 1, 2, 3 . Ходить первыми начинают «крестики». Ни «нолики», ни «крестики» два раза подряд ходить не могут. В случае заполнения 3 полей по вертикали, диагонали или горизонтали одним маркером – выигрыш. Если заведомо понятно, что выстроить 3 в ряд не получится, то, не дожидаясь следующего хода, фиксируется ничья. Необходимо предусмотреть кнопку «перезапуск игры».

19. Револьвер

Револьвер имеет спусковой крючок (курок) и барабан ёмкостью 6 патронов. Барабан желательно реализовать в виде замкнутого списка. При нажатии на курок, если в патроннике (голове списка) есть патрон, происходит выстрел, после чего барабан автоматически поворачивается по часовой стрелке. Если же в патроннике патрона нет, то выстрела не происходит, но барабан всё равно проворачивается по часовой стрелке. Предусмотреть повороты барабана по и против часовой стрелки вручную, а также зарядку патронов в барабан. Начальная конфигурация: барабан пуст.

20. FM–приёмник

Радиовещание на FM–диапазоне лежит в пределах 88–108 МГц. Приёмник содержит кнопки: «+0,25 МГц» и «–0,25 МГц». При попадании частоты настройки приёмника на частоту станции необходимо вывести название станции. Список станций придумать самостоятельно и реализовать в виде массива записей, который является членом класса. Также необходимо предусмотреть кнопку «вкл/выкл питания», которая включает (выключает) приёмник от сети электропитания, причём при включении приёмник настраивается на частоту 88 Мгц. Начальная конфигурация: приёмник выключен.

21. Автоматическая винтовка типа «Штеер»

Винтовка имеет спусковой механизм (курок) и рычаг переключения режимов «одиночный выстрел»/«автоматическая стрельба». При нажатии на курок в зависимости от режима винтовка выстреливает:

* 1 пулю в режиме «одиночный выстрел»;
* 3 пули в режиме «автоматическая стрельба».

Если обойма пуста, то стрельба производиться не может и необходимо сменить обойму. Ёмкость обоймы 20 патронов. Если в обойме осталось менее 3 патронов, то в режиме «автоматическая стрельба» винтовка работать не может. Начальное состояние: в обойме 20 патронов, рычаг переключен в состояние «одиночный выстрел».

22. Автобус типа «ЛиАЗ»

Имеются кнопки управления: «газ» и 3 кнопки «открытие/закрытие дверей» для каждой из дверей. Если двери автобуса закрыты и водитель нажимает «газ», то автобус начинает движение в пределах одной минуты. Если водитель во время движения нажимает «газ», то автобус дополнительно движется одну минуту. Если хотя бы одни из дверей открыты, то автобус с места тронутся не может. Внутри есть освещение которое горит во время остановки и тухнет во время движения (но не во время простоя с закрытыми дверями). Во время движения двери открыться не могут. Если время прошло автобус останавливается. Начальная конфигурация: автобус стоит, двери закрыты.

23. Блок питания

Блок имеет две кнопки регулировки напряжения на выходе «+1 В» и «–1 В», регулятор порогового напряжения 5–15 В с делением 1 В, кнопку «пуск». В блок питания встроена автоматическая защита по напряжению и току. Защита срабатывает:

* если напряжение на выходе будет более порогового;
* если ток в цепи >=5 Ампер;
* если короткое замыкание на выходе (то есть нагрузка равна 0 Ом).

Результатом работы защиты будет отключение напряжения на выходе после нескольких секунд работы в первых 2-х случаях и мгновенное отключение в случае КЗ. Подача напряжения производится с помощью кнопки «пуск». Сила тока рассчитывается по закону Ома. Нагрузка задаётся вручную в Омах. Начальная конфигурация: напряжение на выходе 10В, подключена нагрузка в 3 Ом, пороговое напряжение 15В.

Лабораторная работа №3

**ПРОСТОЕ НАСЛЕДОВАНИЕ И ПОЛИМОРФИЗМ**

**Цель работы:** приобрести навыки в использовании простого наследования и полиморфизма.

**Задание.** Реализовать классы, описанные в вариантах заданий. На примере этих классов продемонстрировать работу механизма позднего связывания.

**Пример.**

**Условие задачи.** Создать базовый класс «эллипс», описав в нем функции ввода параметров, вывода их на экран, нахождения площади эллипса. Породить от него класс «цветной эллипс», переопределив функции ввода и вывода параметров.

//файл classes.h

class Ellipse

{

protected: //a и b должны быть напрямую доступны для класса-потомка

 int a,b;

public:

 Ellipse();

 virtual void EnterParam(); //ввод параметров

 virtual void DisplayParam(); //вывод параметров

 double Area(); //нахождение площади; невиртуальная, так как не переопределяется в классе-потомке

};

class ColorEllipse: public Ellipse

{

 char color[40];

public:

 ColorEllipse();

 virtual void EnterParam();

 virtual void DisplayParam();

};

//файл classes.cpp

#include "classes.h"

#include <stdio.h>

#include <string.h>

#define M\_PI 3.14159265358979323846

Ellipse::Ellipse()

{

 a=1;

 b=1;

}

void Ellipse::EnterParam()

{

 int er;

 do

 {

 printf("Введите длину первой полуоси: ");

 er=scanf("%d",&a);

 if(a<=0 || er==0)

 printf("Ошибка!!! Повторите ввод! \n");

 fflush(stdin);

 }

 while(a<=0 || er==0);

 do

 {

 printf("Введите длину второй полуоси: ");

 er=scanf("%d",&b);

 if(b<=0 || er==0)

 printf("Ошибка!!! Повторите ввод! \n");

 fflush(stdin);

 }

 while(b<=0 || er==0);

}

void Ellipse::DisplayParam()

{

 printf("Длины полуосей: a=%d b=%d \n", a,b);

}

double Ellipse::Area()

{

 return M\_PI\*a\*b;

}

ColorEllipse::ColorEllipse()

{

 strcpy(color,"Красный");

}

void ColorEllipse::EnterParam()

{

 Ellipse::EnterParam();

 printf("Введите цвет эллипса: ");

 scanf("%s",color);

}

void ColorEllipse::DisplayParam()

{

 Ellipse::DisplayParam();

 printf("Цвет эллипса: %s\n",color);

}

//файл mainfile.cpp

#include "classes.h"

#include <stdio.h>

#include <conio.h>

#include <locale.h>

int main()

{

 Ellipse el;

 ColorEllipse colel;

 Ellipse \*pel; //указатель нужен для реализации полиморфизма

 int key1,key2;

 setlocale(0,"RUS"); // обеспесиваем вывод в консольное окно символов кириллицы

 do //выбирвем вид эллипса

 {

 printf("\n1 - обычный эллипс\n");

 printf("2 - цветной эллипс\n");

 printf("ESC - выход\n");

 printf("Ваш выбор: \n");

 key1=getch();

 switch(key1)

 {

 case '1':

 pel=&el;

 break;

 case '2':

 pel=&colel;

 break;

 }

 if (key1!=27)

 {

 do //выбираем действие для выбранного типа эллипса

 {

 printf("\n1 - ввод параметров\n");

 printf("2 - вывод параметров\n");

 printf("3 - посчитать площадь\n");

 printf("ESC - выход\n");

 printf("Ваш выбор: \n");

 key2=getch();

 switch(key2)

 {

 case '1': pel->EnterParam(); break;

 case '2': pel->DisplayParam(); break;

 case '3': printf("Площадь равна: %.2lf\n",pel->Area());

 break;

 }

 }

 while(key2!=27);

 }

 }

 while(key1!=27);

 return 0;

}

**Содержание отчёта**

1. Титульный лист.
2. Условие лабораторной работы.
3. Текст программы.
4. Экранные формы с примерами работы программы.

**Варианты заданий.**

1. Создать базовый класс «треугольник», описав в нём функции задания параметров треугольника, вывода их на экран, нахождения периметра и площади. Породить от него класс «прямоугольный треугольник», переопределив функции задания параметров и нахождения площади. Треугольник задается длинами сторон. При вводе параметров обязательно обеспечить проверку на существование треугольника. При задании параметров для объекта класса «прямоугольный треугольник» проверить, действительно ли задан прямоугольный треугольник.

2. Создать базовый класс «многоугольник», описав в нём функции задания количества и длин сторон, вывода параметров многоугольника на экран, нахождения периметра. Породить от него класс «правильный многоугольник» с числом вершин 3<=N<=6, переопределив в нём функции задания количества и длин сторон и нахождения периметра.

3. Создать базовый класс «отрезок», описав в нем функции задания параметров отрезка, вывода их на экран, нахождения длины отрезка. Породить от него класс «цветной отрезок», переопределив в нем функции задания параметров отрезка и вывода их на экран.

4. Создать базовый класс «треугольник», описав в нём функции задания параметров треугольника, вывода их на экран, нахождения периметра и площади. Породить от него класс «равнобедренный треугольник», переопределив функции задания параметров треугольника и нахождения площади. Треугольник задается длинами сторон. При вводе параметров обязательно обеспечить проверку на существование треугольника.

5. Создать базовый класс «список», определив в нем функции добавления в начало списка, удаления N-го элемента списка, вывода элементов списка на экран. Породить от него класс «упорядоченный список», переопределив функцию добавления элемента списка.

6. Создать базовый класс «таблица» из N\*N полей, определив в нем функции добавления элемента в таблицу, удаление элемента из таблицы по значению, вывод таблицы на экран. Породить от него класс «упорядоченная таблица», переопределив функцию добавления элементов.

7. Создать базовый класс «прямоугольник», определив в нем функции ввода параметров прямоугольника, вывода их на экран и нахождения площади. Породить от него класс «квадрат», переопределив функции ввода параметров и нахождения площади.

8. Создать базовый класс «треугольник», описав в нём функции задания параметров треугольника, вывода их на экран, нахождения периметра и площади. Породить от него класс «правильный треугольник», переопределив функции ввода параметров, нахождения площади и периметра. Треугольник задается длинами сторон. При вводе параметров обязательно обеспечить проверку на существование треугольника.

9. Создать базовый класс «матрица1», описав в нем функции ввода матрицы с клавиатуры, вывода ее на экран и нахождения суммы элементов матрицы. Породить от него класс «матрица2», переопределив в нем функцию ввода матрицы таким образом, чтобы значения элементов матрицы загружались из файла.

10. Создать базовый класс «квадрат1», описав в нем функции ввода параметров квадрата с клавиатуры, вывода их на экран, перемещения квадрата влево и вправо по нажатию клавиш «стрелка влево» и «стрелка вправо». Породить от него класс «квадрат2», переопределив функции нажатия клавиш «стрелка влево» и «стрелка вправо» таким образом, чтобы они выполняли поворот квадрата влево и вправо. Квадрат задается координатой верхнего левого угла и длиной стороны. Выводится на экран должны координаты всех четырех вершин квадрата.

11. Создать базовый класс «массив1», описав в нём функции добавления, удаления, вывода на экран и подсчета количества ненулевых элементов массива. Породить от него класс «массив2», переопределив функцию удаления элемента массива таким образом, чтобы, если удаляемый элемент равен нулю, то обнулялись соседние с ним элементы.

12. Создать базовый класс «массив», описав в нём функции добавления, удаления, вывода на экран и подсчета суммы положительных элементов массива. Породить от него класс «упорядоченный массив», переопределив функцию добавления элементов массива.

13. Создать базовый класс «матрица», описав в нем функции добавления элементов в матрицу и вывода их на экран. Породить от него классы «единичная матрица» и «верхняя треугольная матрица», переопределив в них функцию добавления элементов.

14. Создать базовый класс «треугольник1», описав в нем функции ввода параметров треугольника с клавиатуры, вывода их на экран, перемещения треугольника влево и вправо по нажатию клавиш «стрелка влево» и «стрелка вправо». Породить от него класс «треугольник2», переопределив функции нажатия клавиш «стрелка влево» и «стрелка вправо» таким образом, чтобы они выполняли поворот треугольника влево и вправо. Треугольник задается координатами его вершин.

15. Создать базовый класс «цилиндр», описав в нём функции ввода параметров, вывода их на экран и нахождения объёма цилиндра. Породить от него класс «цилиндр1», который представляет собой фигуру, являющуюся цилиндром, разрезанным плоскостью, проходящей через ось цилиндра. Переопределить в классе «цидиндр1» функцию нахождения объёма фигуры.

16. Создать базовый класс «конус», описав в нём функции ввода параметров, вывода их на экран, нахождения объёма конуса и площади его нижнего основания. Породить от него класс «усеченный конус», переопределив в нем функции ввода и вывода параметров и нахождения объёма фигуры.

17. Создать базовый класс «параллелограмм», определив в нем функции ввода параметров параллелограмма, вывода их на экран и нахождения площади. Породить от него класс «ромб», переопределив функции ввода параметров и нахождения площади.

18. Создать базовый класс «матрица», описав в нем функции добавления элементов в матрицу и вывода их на экран. Породить от него классы «единичная матрица» и «нижняя треугольная матрица», переопределив в них функцию добавления элементов.

19. Создать базовый класс «параллелограмм», описав в нем функции ввода параметров, вывода их на экран, нахождения периметра параллелограмма. Породить от него класс «цветной параллелограмм», переопределив функции ввода и вывода параметров.

20. Создать базовый класс «массив1», описав в нем функции ввода массива с клавиатуры, вывода его на экран и нахождения произведения элементов массива. Породить от него класс «массив2», переопределив в нем функцию ввода массива таким образом, чтобы значения элементов массива загружались из файла.

21. Создать базовый класс «призма», описав функции ввода параметров, вывода их на экран и нахождения объема призмы. Породить от неё класс «параллелепипед», переопределив функции ввода параметров и нахождения объёма.

22. Создать базовый класс «прямоугольник1», описав в нем функции ввода параметров прямоугольника с клавиатуры, вывода их на экран, перемещения прямоугольника влево и вправо по нажатию клавиш «стрелка влево» и «стрелка вправо». Породить от него класс «прямоугольник2», переопределив функции нажатия клавиш «стрелка влево» и «стрелка вправо» таким образом, чтобы они выполняли поворот прямоугольника влево и вправо. Прямоугольник задается координатами верхнего левого и правого нижнего углов. Выводится на экран должны координаты всех четырех вершин прямоугольника.

23. Создать базовый класс «прямоугольник», описав в нем функции ввода параметров, вывода их на экран, нахождения площади прямоугольника. Породить от него класс «цветной прямоугольник», переопределив функции ввода и вывода параметров.

# Лабораторная работа №4

# **Перегрузка операций**

**Цель работы:** приобрести навыки в использовании перегрузки операций.

**Задание:** Согласно условию определить класс, который должен включать в себя метод для ввода данных с обязательной проверкой на корректность вводимых данных, метод вывода данных (обязательна перегрузка операций << и >> для консольного ввода/вывода), метод приведения типа к int или double и указанные в задании перегружаемые операторы. Числа, записанные в строке или в одномерном массиве, должны быть произвольной длины, то есть они не обязательно поместятся в разрядную сетку известных числовых форматов, а, следовательно, для их сложения или вычитания требуется разработка собственного алгоритма.

**Примечание 1.** Методы ++ и – – следует перегрузить как методы класса, а +, –, += и –= можно как методы класса, можно как дружественные функции.

**Примечание 2.** Для методов ++ и – – следует разработать собственный алгоритм добавления или вычитания 1, а для +, –, += и –= можно привести операнды к типу double и выполнить требуемую операцию.

**Пример.**

**Условие задачи.** Создать класс, который запоминает вещественные числа в виде строк. Перегрузить операции \*, и \*=.

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

using namespace std;

class CDoubleString

{

char \*DoubleStr;//указатель на строку, которая хранит число

int count;

public:

 CDoubleString(int n=100) //конструктор

 {

 count=0;

 DoubleStr=new char[n]; //выделение памяти под строку

 }

 CDoubleString(const CDoubleString& c1) //конструктор копирования

 {

 DoubleStr=new char[c1.count+1]; //выделение памяти под строку

 strcpy(DoubleStr,c1.DoubleStr);

 count=c1.count;

 }

 ~CDoubleString()

 {

 if (DoubleStr) delete DoubleStr;

 }

 friend istream & operator>>(istream &, CDoubleString &);//дружественная функция для ввода строки

 friend ostream & operator<<(ostream &, CDoubleString &);//дружественная функция для ввода строки

 void operator \*(const CDoubleString& c1) //перегруженный оператор умножить

 {

 double d1,d2;

 d1=atof(DoubleStr); //перевели строку из текущего в вещественное число

 d2=atof(c1.DoubleStr); //перевели строку из аргумента функции в вещественное число

 d1\*=d2;

 sprintf(DoubleStr,"%f",d1);//перевели число в строку текущего объекта

 count=strlen(DoubleStr)+1; //получили новое значение count

 }

 CDoubleString& operator =(const CDoubleString& c1) //перегруженный оператор присваивания

 {

 if (this!=&c1)

 {

 strcpy(DoubleStr,c1.DoubleStr); //копируем строку из аргумента функции в текущий объект

 count=c1.count;

 }

 return \*this;

 }

 void operator \*=(const CDoubleString& c1)//перегруженный оператор \*=

 {

 double d1,d2;

 d1=atof(DoubleStr);

 d2=atof(c1.DoubleStr);

 d1\*=d2;

 sprintf(DoubleStr,"%f",d1);

 count=strlen(DoubleStr)+1;

 }

 operator double()

 {

 return atof(DoubleStr);

 }

};

istream & operator>>(istream & input, CDoubleString &c1) //определение дружественной функции ввода строки

{

 char key;

 int flag=0; //пользователь не нажал еще десятичную точку

 printf("Enter the number: ");

 do

 {

 key=getch(); //считываем символ с клавиатуры без вывода на акран

 if (c1.count==0 && (key=='-' || (key>='0' && key<='9'))) //проверяем, является ли перевый введенный символ цифрой или минусом

 {

 c1.DoubleStr[c1.count++]=key; //сохраняем введенный символ в строке и увеличиваем счетчик числа введенных символов на 1

 putch(key); //выводим символ на экран

 }

 else

 if (flag==0 && key=='.') //проверяем, первый ли раз нажал пользолватель десятичную точку

 {

 flag=1;//устанавливаем признак введенной десятичной точки

 c1.DoubleStr[c1.count++]=key;

 putch(key);//выводим символ на экран

 }

 else

 if (key>='0' && key<='9') //нажата ли цифра

 {

 c1.DoubleStr[c1.count++]=key;

 putch(key);//выводим символ на экран

 }

 }

 while (c1.count==0 || key!=13); //цикл работает, пока пользователь не нажмет ENTER

 c1.DoubleStr[c1.count++]='\x0';

 printf("\n");

 return input;

}

ostream & operator<<(ostream & output, CDoubleString &c1)

{

 int i;

 i=0;

 cout<<"Number: ";

 while(i<c1.count)

 {

 putch(c1.DoubleStr[i++]);

 }

 printf("\n");

 return output;

}

int main()

{

 CDoubleString d; //создали объект

 cin>>d; //ввели строку

 cout<<d; //вывели строку

 CDoubleString d1;//создали второй объект

 d1=d; //скоировали данные из первого объекта во второй

 cout<<d1; //вывели второй

 d1\*d; //умножили второй на первый, результат во втором

 cout<<d1; //вывели второй

 CDoubleString d2; //создали третий объект

 cin>>d2; //ввели данные в третий объект

 cout<<d2; //выведи третий объект

 d2\*=d; //умножили третий на первый, результат в третьем

 cout<<d2; //вывели третий

 double x;

 x=double(d);

 cout<<x;

 return 0;

}

**Содержание отчёта**

1. Титульный лист.
2. Условие лабораторной работы.
3. Текст программы.
4. Экранные формы с примерами работы программы.

**Варианты заданий.**

1. Создать класс, который запоминает целые десятичные числа в виде строк. Перегрузить операции +, ++ и +=.
2. Создать класс, который запоминает целые десятичные числа в виде строк. Перегрузить операции –, – – и –=.
3. Создать класс, который запоминает вещественные числа в виде строк. Перегрузить операции +, ++ и +=.
4. Создать класс, который запоминает вещественные числа в виде строк. Перегрузить операции –, – – и –=.
5. Создать класс, который запоминает целые двоичные числа в виде строк. Перегрузить операции +, ++ и +=.
6. Создать класс, который запоминает целые двоичные числа в виде строк. Перегрузить операции –, – – и –=.
7. Создать класс, который запоминает целые восьмеричные числа в виде строк. Перегрузить операции +, ++ и +=.
8. Создать класс, который запоминает целые восьмеричные числа в виде строк. Перегрузить операции –, – – и –=.
9. Создать класс, который запоминает целые шестнадцатеричные числа в виде строк. Перегрузить операции +, ++ и +=.
10. Создать класс, который запоминает целые шестнадцатеричные числа в виде строк. Перегрузить операции –, – – и –=.
11. Создать класс, который запоминает целые десятичные числа в виде одномерного массива. Перегрузить операции +, ++ и +=.
12. Создать класс, который запоминает целые десятичные числа в виде одномерного массива. Перегрузить операции –, – – и –=.
13. Создать класс, который запоминает целые двоичные числа в виде одномерного массива. Перегрузить операции +, ++ и +=.
14. Создать класс, который запоминает целые двоичные числа в виде одномерного массива. Перегрузить операции –, – – и –=.
15. Создать класс, который запоминает целые шестнадцатеричные числа в виде одномерного массива. Перегрузить операции +, ++ и +=.
16. Создать класс, который запоминает целые шестнадцатеричные числа в виде одномерного массива. Перегрузить операции –, – – и –=.
17. Создать класс, который запоминает целые восьмеричные числа в виде одномерного массива. Перегрузить операции +, ++ и +=.
18. Создать класс, который запоминает целые восьмеричные числа в виде одномерного массива. Перегрузить операции –, – – и –=.
19. Создать класс, который запоминает вещественные числа в виде одномерного массива. Перегрузить операции +, ++ и +=.
20. Создать класс, который запоминает вещественные числа в виде одномерного массива. Перегрузить операции –, – – и –=.
21. Создать класс, который запоминает римские числа в виде строк. Перегрузить операции +, ++ и +=.
22. Создать класс, который запоминает римские числа в виде строк. Перегрузить операции –, – – и –=.

# **Лабораторная работа №5**

# **шаблоны классов**

**Цель работы:** приобрести навыки в создании и использовании шаблонов классов.

**Задание:** определить шаблон класса CList, который должен включать в себя следующие методы: добавление элемента в список, вывод списка на экран, освобождение памяти, занимаемой списком, конструктор, деструктор, а также метод, указанный в вариантах заданий. Для описания методов шаблона класса использовать внешнюю реализацию. В функции main() создать объекты CList для элементов типа int, double, char\* и объектов класса, определенного в лабораторной работе №1, дополнив его перегруженной операцией вывода массива (<<). Для этих объектов продемонстрировать работу всех методов шаблона класса CList.

**Примечание**. Метод, указанный в вариантах заданий, должен возвращать логическое значение: **true**, если удалось выполнить требуемое удаление (вставку) элементов, и **false** – если не удалось.

**Пример.**

**Условие задачи.** Добавить в шаблон класса CList метод, удаляющий первое вхождение элемента со значением d.

template<class type>

class CListTemplate

{

 struct SNode

 {

 type \*data;

 SNode \*next;

 };

 SNode \*head,\*p;

public:

 CListTemplate() {head=p=0;}

 void AddElem(type \*d);

 bool RemoveElem(type \*d);

 void DisplayList();

 void clear();

 ~CListTemplate(){clear();}

};

template<class type>

void CListTemplate<type>::AddElem(type \*d)

{

 if(head)

 {

 p->next = (SNode\*)malloc(sizeof(SNode));

 p = p->next;

 }

 else

 {

 head = (SNode\*)malloc(sizeof(SNode));

 p=head;

 }

 p->data = d;

 p->next = 0;

}

template<class type>

bool CListTemplate<type>::RemoveElem(type \*d)

{

 SNode \*curr = head;

 SNode \*p1;

 if (\*(head->data)==\*d)

 {

 p1=head;

 head=head->next;

 delete p1;

 return true;

 }

 while(curr->next && \*(curr->next->data)!=\*d)

 curr=curr->next;

 if (curr->next)

 {

 p1=curr->next;

 curr->next=curr->next->next;

 delete p1;

 return true;

 }

 return false;

}

template<class type>

void CListTemplate<type>::DisplayList()

{

 SNode \*curr=head;

 while (curr)

 {

 cout<<\*(curr->data)<<" ";

 curr=curr->next;

 }

 cout<<endl;

}

template<class type>

void CListTemplate<type>::clear()

{

 SNode \*curr=head;

 while(head)

 {

 curr=head;

 head=head->next;

 delete curr;

 }

}

int main()

{

 CListTemplate<Complex> cl;

 CListTemplate<int> c\_int;

 for (int i=0;i<10; i++)

 {

 cl.AddElem(new Complex(10.0f\*i,10.0f+i));

 c\_int.AddElem(new int(10\*i));

 }

 cout<<endl<<endl<<endl;

 cl.DisplayList();

 Complex c\_ob(40.0f,14.0f);

 cl.RemoveElem(&c\_ob);

 cl.DisplayList();

 cout<<endl<<endl<<endl;

 c\_int.DisplayList();

 int i=90;

 c\_int.RemoveElem(&i);

 c\_int.DisplayList();

 return 0;

}

**Содержание отчёта**

1. Титульный лист.
2. Условие лабораторной работы.
3. Текст программы.
4. Экранные формы с примерами работы программы.

**Варианты заданий.**

1. Добавить в шаблон класса CList метод, удаляющий из списка последние N элементов.
2. Добавить в шаблон класса CList метод, удаляющий из списка первые N элементов.
3. Добавить в шаблон класса CList метод, удаляющий из списка все элементы, стоящие на четных позициях. Нумерацию элементов списка начинать с 0.
4. Добавить в шаблон класса CList метод, удаляющий из списка все элементы, стоящие на нечетных позициях. Нумерацию элементов списка начинать с 0.
5. Добавить в шаблон класса CList метод, удаляющий из списка каждый N–й по счету элемент.
6. Добавить в шаблон класса CList метод, меняющий местами первый и последний элемент списка.
7. Добавить в шаблон класса CList метод, дублирующий в списке все элементы со значением N.
8. Добавить в шаблон класса CList метод, дублирующий в списке каждый N–й по счету элемент.
9. Добавить в шаблон класса CList метод, удаляющий из списка все элементы со значением N.
10. Добавить в шаблон класса CList метод, инвертирующий элементы списка относительно их исходного положения. Например, исходный список: <1, 2, 3, 4>; результирующий список: <4, 3, 2, 1>
11. Добавить в шаблон класса CList метод, циклически сдвигающий список вправо на N разрядов. Например, список <1, 2, 3, 4> необходимо сдвинуть на 2 разряда; результирующий список: <3, 4, 1, 2>.
12. Добавить в шаблон класса CList метод, циклически сдвигающий список влево на N разрядов. Например, список <1, 2, 3, 4> необходимо сдвинуть на 1 разряд; результирующий список: <2, 3, 4, 1>.
13. Добавить в шаблон класса CList метод, вставляющий за элементом с номером N элемент со значением Y. Нумерацию элементов списка начинать с 1.
14. Добавить в шаблон класса CList метод, вставляющий после первого вхождения элемента со значением X элемент со значением Y.
15. Добавить в шаблон класса CList метод, удаляющий элемент, расположенный в списке сразу за первым вхождением элемента со значением X.
16. Добавить в шаблон класса CList метод, вставляющий перед первым вхождением элемента со значением X элемент со значением Y.
17. Добавить в шаблон класса CList метод, удаляющий элемент, расположенный в списке перед первым вхождением элемента со значением X.
18. Добавить в шаблон класса CList метод, вставляющий перед элементом с номером N элемент со значением Y. Нумерацию элементов списка начинать с 1.
19. Добавить в шаблон класса CList метод, вставляющий за элементом с номером N k элементов со значением Y. Нумерацию элементов списка начинать с 1.
20. Добавить в шаблон класса CList метод, удаляющий k элементов, расположенных в списке после элемента с номером N. Нумерацию элементов списка начинать с 1.
21. Добавить в шаблон класса CList метод, вставляющий за первым вхождением элемента со значением X k элементов со значением Y.
22. Добавить в шаблон класса CList метод, удаляющий после первого вхождения элемента со значением X k элементов.
23. Добавить в шаблон класса CList метод, меняющий местами максимальный и минимальный по значению элемент списка.
24. Добавить в шаблон класса CList метод, дублирующий каждый элемент списка.
25. Добавить в шаблон класса CList метод, вставляющий перед первым вхождением элемента со значением X элемент со значением Y, а за элементом X – элемент со значением Z.
26. Добавить в шаблон класса CList метод, удаляющий элементы списка, расположенные перед и за первым вхождением элемента со значением X.
27. Добавить в шаблон класса CList метод, удаляющий элементы списка, расположенные перед и за элементом с номером N. Нумерацию элементов списка начинать с 1.
28. Добавить в шаблон класса CList метод, вставляющий перед элементом списка c номером N элемент со значением X, а после – элемент со значением Y. Нумерацию элементов списка начинать с 1.
29. Добавить в шаблон класса CList метод, вставляющий перед элементом с номером N k элементов со значением Y. Нумерацию элементов списка начинать с 1.
30. Добавить в шаблон класса CList метод, вставляющий перед первым вхождением элемента со значением X k элементов со значением Y.