ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

И.М. Егоров

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА C++

РУКОВОДСТВО ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ СПЕЦИАЛЬНОСТИ 210106

ТОМСК — 2007

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра промышленной электроники (ПРЭ)

И.М. Егоров

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА C++

РУКОВОДСТВО ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ СПЕЦИАЛЬНОСТИ 210106

2007

Егоров И.М.
Объектно-ориентированное программирование на C++: Руководство по организации самостоятельной работы студентов специальности 210106. — Томск: Томский государственный университет систем управления и радиоэлектроники, 2007. — 47 с.

Пособие предназначено для студентов специальности 210106 «Промышленная электроника», изучающих дисциплину «Объектно-ориентированное программирование» и выполняющих лабораторный практикум по данной дисциплине.

Пособие содержит полный текст утвержденной рабочей программы курса, перечень вопросов для контрольных работ, формулировку требований к выполнению лабораторных работ, их описание и рейтинговую раскладку.

СОДЕРЖАНИЕ

РАБОЧАЯ ПРОГРАММА КУРСА «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

Федеральное агентство по образованию

ТОМСКИЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

УТВЕРЖДАЮ

Проректор по учебной работе

_____М.Т. Решетников

"21" 04 2006 года

РАБОЧАЯ ПРОГРАММА

По дисциплине: «Объектно-ориентированное программирование»

Для специальности 210106 «Промышленная электроника»

Направление 210100 «Электроника и микроэлектроника»

Факультет: Электронной техники

Профилирующая кафедра: Промышленной электроники

Курс: четвертый

Семестр: седьмой

Учебный план набора 2003 года и последующих лет

Распределение учебного времени	Всего часов
Лекции (7 семестр)	36
Лабораторные (7 семестр)	54
Всего аудиторных занятий	90
Самостоятельная работа	90
Общая трудоемкость	180
Экзамен	7 семестр

2006

Рабочая программа составлена на основании ГОС ВО для направления 210100 «Электроника и микроэлектроника» № 22 тех/дс, утвержденного 10.03.2000, рассмотрена и утверждена на заседании кафедры «___»_____200___года, протокол № _____

Разработчик
доцент каф. ПрЭ И.М. Егоров

Зав. выпускающей
кафедрой ПрЭ А.В. Кобзев

Рабочая программа согласована с факультетом
Декан ФЭТ В.М. Герасимов

1 ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

1.1 Цель преподавания дисциплины

Целью преподавания дисциплины «Объектно-ориентированное программирование» является изучение основ современной технологии программирования с использованием агрегированных объектных типов данных и реализации этой технологии средствами алгоритмического языка С++.

1.2 Задачи изучения дисциплины

В результате изучения дисциплины «Объектно-ориентированное программирование» студент получает представление о производительных методах синтеза прикладных программ с использованием классов С++, овладевает навыками построения систем классов для решения собственных инженерных задач, что значительно сокращает временные затраты при создании программного обеспечения, повышает преемственность программного продукта при его доработке и перепрофилировании.

1.3 Связь с другими дисциплинами

Изучение дисциплины «Объектно-ориентированное программирование» базируется на курсах информатики и программирования на языке С++. В свою очередь, материал курса «Объектно-ориентированное программирование» служит базой для освоения курсов программирования в интегрированных средах разработки Borland C++ Builder, Microsoft Visual C и др.

2 СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1 Возникновение и развитие объектно-ориентированного подхода как технологии программирования (2 часа)

Понятие о технологии программирования. Необходимость технологических решений при создании программного продукта. Развитие средств автоматизации программирования и его техно-

логии. Формирование общих принципов построения программ: модульность, скрытие (инкапсуляция) данных, интерфейсы взаимодействия. Типизация языка программирования. Наследование типов как средство преемственности разработок. Основные концепции объектно-ориентированного подхода: абстракция, инкапсуляция, наследование, полиморфизм.

2.2 Объекты и типы программы, области видимости и время жизни объектов (2 часа)

Предопределенные типы и типы, определяемые пользователем. Введение типов и объектов в программу. Основные и производные типы. Области видимости объектов. Скрытие объектов и расширение контекста. Время жизни и классы хранения. Классы хранения **auto, register**, **static, extern**. Единицы трансляции и компоновка. Пространство имен функций. Область видимости имен функций. Инициализация глобальных и локальных переменных.

2.3 Работа с указателями и ссылками. Адресная арифметика C++ (2 часа)

Операции с указателями. Контроль за адресной арифметикой. Указатель на **void**. Инициализация указателей и ссылок. Значение указателя **NULL**. Доступ к элементам массива через указатель. Доступ к элементам объектных типов по указателю. Многомерные массивы, массивы указателей и указатели на указатели. Динамическое создание объектов.

2.4 Функциональная декомпозиция задачи и процедурное программирование (2 часа)

Функциональный подход в программировании и модульный принцип построения исходного кода программ. Функции C++ — средство модульного программирования. Формат обмена информацией между функцией и внешней средой. Синтаксис записи интерфейса функции: возвращаемое значение, имя функции, список формальных параметров. Функции без возврата значения. Функция **main** — головная программа. Передача параметров функции **main**.

2.5 Функций C++: описание и вызов (3 часа)

Функция как именованный блок с интерфейсом. Структура интерфейса функции. Объявление функции (прототип). Исходные файлы и объявление переменных и функций. Заголовочные **h**-файлы и директива препроцессора **#include**. Определение функции. Оператор вызова функции. Прием и передача информации через интерфейс. Контроль типов параметров, принимаемых функцией при вызове и возможности их автоматического приведения к интерфейсу. Задание параметров по умолчанию. Функции с переменным числом параметров. Идентификация функций компилятором C++, понятие сигнатуры функции. Перегрузка функций. Эквивалентность функций и операторов в C++, перегрузка операторов. Указатели на функции и работа с ними.

2.6 Функции C++: обмен информацией с вызывающей программой (3 часа)

Инкапсуляция данных внутри функции. Особенности использования внутренних элементов функций с различными классами хранения. Порядок передачи параметров по списку, модификатор **pascal**. Роль стека в передаче параметров. Передача параметров по значению по ссылке. Использование имен массивов, указателей и ссылок в качестве формальных параметров. Побочные эффекты и их использование в информационном обмене. Блокировка нежелательных побочных эффектов.

2.7 Внутренняя реализация кода функций (2 часа)

Особенности реализации программного кода функции на низком уровне. Реализация функций со стандартным интерфейсом. Функции **inline**. Функции обработки прерываний. Возможности программирования на низком уровне. Встроенный ассемблер. Регистровые псевдопеременные Borland C++.

2.8 Объектно-ориентированные средства C++. Объектные типы данных. Протокол класса C++ (2 часа)

Виды объектных данных: структуры, объединения, классы. Синтаксис описания объектного типа. Описание протокола клас-

са. Отношения «класс — экземпляр класса» и «тип — объект». Поля и методы класса. Протокол класса. Регламентация доступа к компонентам класса: частные (**private**), защищенные (**protected**) и общие (**public**) компоненты. Доступ к данным через объект и через протокол класса. Область видимости данных в классах. Синтаксис определения функции — элемента в протоколе класса и вне его. Статические элементы класса и особенности их использования.

2.9 Конструкторы и деструкторы объектных типов. Инициализация объектов в исполняемой программе (2 часа)

Конструктор объектных данных как функция — элемент класса, его назначение, синтаксис описания и вызова. Конструктор без параметров (void — конструктор). Конструктор копирования. Автоматический и явный вызов конструктора. Инициализация полей объекта в конструкторе. Использование динамического распределения памяти при создании объектов. Перегружаемые конструкторы. Деструктор объекта. Автоматический вызов деструктора. Необходимость в явном определении деструктора.

2.10 Перегрузка функции и операции как реализация полиморфизма. Дружественные функции и классы (3 часа)

Зависимость реакции объекта от сигнатуры функции. Задание формальных параметров по умолчанию. Возможная неоднозначность в сигнатуре при вызове методов с параметрами по умолчанию. Понятие дружественных функций. Синтаксис переопределения операции. Переопределение одно- и двуместных операций в методах класса. Указатель ***this*** в методах класса. Объявление дружественных операций.

2.12 Наследование классов C++ как основа развития технологии программирования (3 часа)

Определение наследования, базовые и производные классы, иерархия классов. Единичное и множественное наследование. Спецификаторы типа наследования. Доступ к элементам базового

класса в потомках, роль защищенных компонент классов. Доступ к элементам объектов производных классов через указатель на базовый объект. Особенности перегрузки функций при наследовании. Виртуальные функции. Понятие о раннем и позднем связывании. Чистые виртуальные функции и абстрактные классы. Виртуальные классы при множественном наследовании.

2.13 Шаблоны функций и классов. Параметризация типов данных (2 часа)

Однородные операции над различными типами данных. Формальное определение типа и шаблоны функций. Синтаксис объявления шаблона функции. Родовые классы и шаблоны классов. Синтаксис объявления шаблона класса. Построение системы классов на основе шаблонов. Использование шаблонов при определении типов. Шаблоны классов и дружественные функции.

2.14 Потоковые классы (4 часа)

Потоки данных. Источники и приемники. Концепция буферизации данных при обмене. Управление потоком. Состояние потока, включение в поток, извлечение из потока, Система потоковых классов библиотеки Borland C. Базовые классы ios и streambuf. Специализация потоков. Консольные потоки: классы constream и conbuf. Файловые потоки: классы fstream и filebuf. Потоки «память — память»: классы strstream и strbuf. Реализация процедур ввода/вывода с помощью потоков.

2.15 Контейнерные классы (4 часа)

Система абстрактных классов для хранения данных объектных данных. Контейнерные классы. Два способа физического распределения памяти для контейнера: вектор и список. Массив списков (Хэш-таблица). Прямые и косвенные контейнеры. Сортированные и несортированные контейнеры. Фундаментальные структуры данных (FSD) и абстрактные типы данных. Объектно-ориентированная библиотека контейнерных классов и библиотека шаблонных классов BIDS.

3 ПЕРЕЧЕНЬ ЛАБОРАТОРНЫХ РАБОТ

№	Содержание лабораторной работы	Часы	Максимальный рейтинговый балл
1	Программа на С++ работы с трехмерными векторами. Составление функций	4	3
2	Программа на С++, реализующая функцию ввода символов с клавиатуры с фильтрацией информационных и управляющих символов.	3	4
3	Программа на С++ для работы с матрицами с интерфейсом ввода. Составление функций интерфейса	3	5
4	Работа со структурами, объединениями, битовыми полями и поразрядными операциями	4	7
5	Программирование динамических списков. Сортировка и фильтрация	4	7
6	Протокол класса для описания трехмерных векторов как математических объектов	4	4
7	Построение класса для математического объекта «матрица» с использованием динамического распределения памяти	4	7
8	Исследование модификаций доступа к данным в производных классах при одиночном наследовании. Реализация механизма позднего связывания. Виртуальные функций в иерархии классов	4	8
9	Построение класса для описания объекта «дата/время». Перегрузка операций сложения и вычитания. Реализация методов «день недели» и «текущая дата и время»	4	8
10	Построение класса для описания объекта «обыкновенная дробь» с перегрузкой арифметических операций, операций отношения и операций потокового вывода. Создание методов сокращения дроби и выделения целой части дроби	4	6
11	Построение класса для описания объекта «полином с вещественными коэффициентами». Реализация арифметических действий над полиномами: сложения, вычитания, умножения, деления с остатком	4	11
12	Построение шаблонного класса для описания объекта «полиномиальная дробь», реализация арифметики полиномиальных дробей	4	10
Всего		**46**	**80**

4 КОНТРОЛЬНЫЕ РАБОТЫ

(выполняются за счет лимита времени лабораторных работ)

№	Тематика контрольной работы	Часы	Максимальный рейтинговый балл
1	Основные концепции объектно-ориентированного программирования. Синтаксис определения объектного типа. Доступ к элементам объектных типов	2	10
2	Перегрузка функций и операторов в C++. Сигнатура функций. Маскировка имен и разрешение контекста	2	10
3	Наследование классов. Виртуальные функции. Абстрактные базовые классы. Виртуальные классы при множественном наследовании	2	10
	Всего	**6**	**30**

5. Индивидуальное собеседование (2 часа)

Индивидуальное собеседование проводится в устной форме после завершении цикла лабораторных занятий по материалам представленных исходных текстов программ. В ходе собеседования студент должен без дополнительной подготовки прокомментировать работу любого фрагмента представленной им программы. Максимальный рейтинговый балл — 10.

6. Рейтинговая раскладка дисциплины:

Лабораторный практикум	80
Контрольные работы	30
Индивидуальное собеседование	10
	120

7. Самостоятельная работа студентов

№ пп	Содержание работы	Объем, часов	Форма контроля
1. 2.	Подготовка к лекционным занятиям Подготовка к лабораторным работам	36 36	Контрольные работы Защита лабораторных работ
3	Подготовка к контрольным работам	18	Проверка контрольных работ
	Всего	90	

8 ЛИТЕРАТУРА

8.1 Основная литература

1. Франка П. C++: Учебный курс: Пер. с англ. П. Бибикова. — СПб.: Питер, 2005. — 521с

2. Павловская Т.А., Щупак Ю.А. C++. Объектно-ориентированное программирование: практикум: учебное пособие для вузов. — СПб.: Питер, 2005. — 464 с

3. Павловская Т.А. C/C++: Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2002. — 460 с

4. Боровский А.Н. Самоучитель C++ и Borland C++ Builder: самоучитель. — СПб.: Питер, 2005. — 255 с.

8.2 Дополнительная литература

1. Ричард Вайнер, Льюис Пинсон. C++ изнутри: Пер. с англ. — Киев: «ДиаСофт», 1993. — 304 с.

2. Стефан Дьюхарст, Кэти Старк. Программирование на C++. Пер. с англ. — Киев: «ДиаСофт», 1993. — 272 с.

3. Ирэ Пол. Объектно-ориентированное программирование с использованием C++: Пер. с англ. — Киев: «ДиаСофт» ,1995. — 480 с.

4. Фейсон Т. Объектно-ориентированное программирование на Borland C++ 4.: Пер. с англ. — Киев: Диалектика, 1996. — 544 с.

5. Скляров В.А. Язык C++ и объектно-ориентированное программирование: Справочное издание. — Минск: Вышэйшая школа, 1997. — 480 с.

6*. Бьярн Страуструп Введение в C++. 1995
http://www.umx.org.ua/cpp/aglav.htm

7*. Патрикеев Ю.Н. Объектно-ориентированное программирование на BORLAND C++: Лекции. 1998
http://www.object.newmail.ru/obj0.html

8*. Вахтеров М., Орлов С. Четвертый BORLAND C++ и его окружение, http://www.unix.org.ua/bccp/index.htm

Примечание: символом * помечены электронные учебники.

ВОПРОСЫ НА КОНТРОЛЬНЫЕ РАБОТЫ

1. Что такое объектный тип в C++? Какие зарезервированные слова (лексемы языка) используются для их описания?

2. Что такое конструктор и деструктор класса? Каковы их функции?

3. Что обозначают следующие директивы:

```
double a;
double *a;
double &a;
```

4. Что такое конструктор по умолчанию и конструктор копирования, в каких случаях происходит их неявный вызов (вызов самим компилятором)?

5. Что такое передача параметров по значению и по ссылке? Какова между ними разница?

6. Дайте определение области видимости объектов. Что означает термин «время жизни (существования)» объекта? Что такое скрытие (маскировка) имен и когда это явление наблюдается?

7. В чем состоит сущность перегрузки функций и операторов в C++? Что называется полиморфизмом в объектно-ориентированном программировании?

8. Что называют протоколом класса? Каков синтаксис протокола? Каково назначение регламентации доступа к элементам класса, в чем смысл меток **public, protected** и **private**?

9. Поясните смысл и назначение поля **this**. Что в нем хранится?

10. Что называют элементами класса? В чем особенность функций — элементов, какими дополнительными «правами» они наделены?

11. В чем состоит побочный эффект функции? Какова основная (используемая по умолчанию) схема передачи информации внутрь функции и из функции в вызывающую программу?

12. Виртуальные функции и позднее связывание. Что это такое?

13. Что такое «чистая» виртуальная функция? Какова особенность классов, содержащих чистые виртуальные функции?

14. Укажите различия между объявлением и определением функции. Каков синтаксис определения функции — элемента класса за пределами его протокола?

15. Что такое «дружественные» функции? Каков синтаксис объявления дружественных функций в классе?

16. Что такое «дружественные классы»? Как они объявляются?

17. Механизм наследования классов. Отношение классов «базовый/производный». Что называют единичным и множественным наследованием?

18. Что обозначают следующие выражения **VV.m** и **UU ->m** ? Когда они используются?

19. В чем отличие объектных типов, описанных протоколом с ключевыми словами **class** и **struct**?

20. Что такое **inline**-функции, в чем их отличие от обычных функций, когда возможно и целесообразно использовать **inline**-функцию?

21. Что такое статические элементы класса? В чем различие статических и нестатических полей класса?

22. В каком соотношении находятся области видимости базового и производного классов?

23. Что называют абстрактным классом? Для чего используются абстрактные классы?

ПОРЯДОК АТТЕСТАЦИИ ПО ЛАБОРАТОРНОМУ ПРАКТИКУМУ

Выполнение лабораторных работ допустимо по бригадному принципу, когда одна программа создается группой не более, чем из 2–3 человек, но зачет по лабораторным работам производится по результатам ее защиты, проводимой в форме *индивидуального* собеседования. На защиту принимается полностью отлаженная и, безусловно, работоспособная программа.

В процессе защиты лабораторной работы *каждый* студент должен быть готов дать комментарии по любому фрагменту защищаемой программы. С целью упрощения чтения программы ее текст целесообразно снабдить необходимыми комментариями.

Оценка работ производится по рейтинговой системе: каждой работе сопоставлен максимальный рейтинговый балл, начисляемый в случае ее успешной защиты. Рейтинговая раскладка лабораторных работ приведена в следующей таблице:

№	Содержание лабораторной работы	Максимальный рейтинговый балл
1	Программа на C++ работы для работы с трехмерными векторами. Составление функций	3
2	Программа на C++, реализующая функцию ввода символов с клавиатуры с фильтрацией информационных и управляющих символов	4
3	Программа на C++ для работы с матрицами с интерфейсом ввода. Составление функций интерфейса	5
4	Работа со структурами, объединениями, битовыми полями и поразрядными операциями	7
5	Программирование динамических списков. Сортировка и фильтрация	7
6	Протокол класса для описания трехмерных векторов как математических объектов	4
7	Построение класса для математического объекта «матрица» с использованием динамического распределения памяти	7
8	Исследование модификаций доступа к данным в производных классах при одиночном наследовании. Реализация механизма позднего связывания. Виртуальные функций в иерархии классов	8

№	Содержание лабораторной работы	Максимальный рейтинговый балл
9	Построение класса для описания объекта «дата/время». Перегрузка операций сложения и вычитания. Реализация методов «день недели» и «текущая дата и время»	8
10	Построение класса для описания объекта «обыкновенная дробь» с перегрузкой арифметических операций, операций отношения и операций потокового вывода. Создание методов сокращения дроби и выделения целой части дроби	6
11	Построение класса для описания объекта « полином с вещественными коэффициентами». Реализация арифметических действий над полиномами: сложения, вычитания, умножения, деления с остатком	11
12	Построение шаблонного класса для описания объекта « полиномиальная дробь», реализация арифметики полиномиальных дробей	10
Всего		**80**

Таким образом, выполнив и защитив все работы, студент может набрать 80 рейтинговых баллов, что соответствует оценке «хорошо» за экзамен. Если работы выполняются в высшей степени самостоятельно, их защита проходит на хорошем уровне и в срок, то рейтинговый балл в индивидуальном порядке может быть увеличен до 25 % от указанного в таблице.

Если студент претендует на более высокую оценку, чем исчисляемая по рейтингу, то он может пройти индивидуальное собеседования по материалам всех выполненных им работ и получить дополнительные 25 % баллов, при условии, что сумма баллов при этом не превысит 120.

Оценка качества программы зависит не столько от факта ее работоспособности, сколько от того, насколько качественно написан ее исходный код. Популярный среди студентов аргумент: «ведь и так работает же!», здесь не действует.

Внимание! Для решения возможных спорных вопросов, а также для прохождения заключительного собеседования, сохраняйте файлы с исходными текстами программ до конца семестра.

ОБЩАЯ ХАРАКТЕРИСТИКА ЛАБОРАТОРНЫХ РАБОТ

Лабораторный практикум по курсу «Объектно-ориентированное программирование» включает в себя 12 лабораторных работ. Характер предлагаемых заданий рассчитан на студентов, получивших элементарные сведения о программировании на языке C++ и некоторый опыт практической работы с этим языком.

Содержание практикума — решение задач с применением двух технологий программирования: *процедурной* и *объектной*. Каждая из них предусматривает декомпозицию исходной задачи и представление ее решения как результата взаимодействия группы относительно независимых программных объектов. В процедурно-ориентированном программировании роль таких элементарных объектов играют *подпрограммы — функции C++*, при объектно-ориентированной технологии — это *объекты, созданные на основе классов C++*.

При освоение процедурно-ориентированного подхода (лабораторные работы №1 — №5) у студента должен сформироваться навык функциональной декомпозиции задачи и реализации программы на языке C++ по модульному принципу с использованием функций. Основные требования здесь — максимальная структуризация программы, выражающаяся в том, что функции должны:

– решать только ту задачу, для которой они сформированы;

– обеспечивать обмен информацией только через свой интерфейс, *использование глобальных переменных не допустимо*;

– быть некритичными к размерам массивов, если таковые используются в качестве фактических параметров.

При решении задач на основе объектно-ориентированного программирования (лабораторные работы №6 — №12) следует произвести структурную декомпозицию задачи, выяснить, какие объекты и в каком взаимодействии друг с другом составляют сущность задачи. На основе такого анализа следует обоснованно составить описание класса, наделив его необходимыми атрибутами, выделить в нем элементы с различным регламентом доступа, предусмотреть необходимый набор конструкторов, выяснить необходимость деструктора. Создав на основе классов объекты, необходимо ввести их в программу. Определение функций — элементов класса следует выполнять вне протокола класса. Пере-

дачу функциям аргументов объектного типа следует по ссылке, при необходимости защитив их от возможного изменения значений внутри функции.

Практически ни одна программа на C++ не может обойтись без библиотечных функций. При использовании библиотечных функций не забывайте подключать необходимые заголовочные файлы (h-файлы) с их объявлениями.

ЛАБОРАТОРНАЯ РАБОТА № 1
РАБОТА С ТРЕХМЕРНЫМИ ВЕКТОРАМИ

Цель работы. Ознакомление с правилами составления элементарных программ на языке C++ с использованием функций.

Тематика изучаемого материала: функции и их интерфейсы, передача параметров функциям и возврат информации в вызывающую программу.

Задание к лабораторной работе

Составить программу для работы с векторами, задаваемыми координатами в трехмерном пространстве. Вектора задать массивами элементов типа **float** размера 3. Процедуры обработки векторов оформить в виде функций C++.

1. С помощью функций обеспечить следующие одноместные операции:

- доступ к элементам вектора (чтение/запись);
- вычисление модуля вектора;
- умножение вектора на скаляр;
- нормировку вектора (получение вектора единичной длины) .

2. С помощью функций обеспечить двуместные операции над векторами A и B:

а) с получением нового вектора C:

- сложение (C = A + B);
- вычитание (C = A – B);
- векторное произведение (C = A * B);

б) с получением скалярных величин:

- скалярного произведения двух векторов;
- косинуса и синуса угла между двумя векторами (угол в пределах от 0 до 180 град);
- величины угла в градусах между векторами в пределах [0,180].

Указание: для расчета угла воспользуйтесь функцией **double** `acos(` **double** `x)`, подключив заголовочный файл **math.h**

Создайте две функции вывода на экран компонент вектора в удобной форме, например, в виде строки:

x=1.67 y=54.95 z=17.6

Для организации ввода/вывода в первой функции воспользуйтесь средствами потокового ввода/вывода (**cin >> и cout <<**), подключив предварительно заголовочный файл **iostream.h.**

Во второй функции следует применить библиотечные функции форматированного ввода/вывода **scanf** и **printf**

Сведения, необходимые для программирования методов векторной алгебры представлены в Приложении А.

ЛАБОРАТОРНАЯ РАБОТА № 2
ПРОГРАММА ВВОДА ВЕЩЕСТВЕННОГО ЧИСЛА И УПРАВЛЯЮЩИХ СИМВОЛОВ

Цель работы. Получение навыков управления программой с помощью ввода символов с клавиатуры.

Тематика изучаемого материала: операторы ветвления вычислительного процесса, циклы и логические выражения.

Задание к лабораторной работе

Составить программу, осуществляющую обработку потока символов ASCII с клавиатуры с разделением их на три категории:

1. информационные,
2. управляющие,
3. неинформационные.

Нужно организовать разную реакцию программы на ввод символов различных категорий.

Информационные символы, должны в порядке поступления накапливаться в символьном массиве — буфере, откуда в определенный момент направляются на дальнейшую обработку в виде сплошной символьной строки.

Управляющие символы (4–5 заданных символов ASCII) используются для ветвления вычислительного процесса. В частности, ввод одного из них может инициировать обработку накопленной в буфере информации, ввод другого — вызывать завершение программы и т.д.

Неинформационные символы должны игнорироваться программой и не вызывать в ней никакой реакции.

Обработка символьной информации буфера сводится к преобразованию содержащейся в нем символьной строки в вещественное число с помощью библиотечной функции с именем **atof**. Описание прототипа этой функции содержится в заголовочном файле **math.h**, который следует подключить с помощью директивы препроцессору #include <math.h>. Прототип имеет вид:

```
double atof (const char*)
```

Эта функция принимает аргументом строку символов, соответствующую внешнему представлению числа и возвращает значение вещественного числа во внутреннем формате компьютера.

Например запись:

```
double d = atof("-12.678");
```

приведет к инициализации переменной d значением — 12.678

Таким образом, к информационным символам, пропускаемым в буфер, следует отнести только те символы, которые образуют внешнее представление вещественного числа. Это коды символов минус(–), точка (.) и коды десятичных цифр (от 0 до 9). Прочие символы не должны заноситься в буфер.

При вводе необходим логический контроль последовательности заносимых в буфер информационных символов. Так, символ минус может появиться в буфере один раз, занимая в нем только лидирующее положение, символ точки тоже может заноситься в буфер только однажды.

После обработки символьной последовательности буфера, он должен быть очищен и готов к приему следующей последовательности. Перед отправкой содержимого буфера на обработку его необходимо дополнить т.н. «нуль терминатором» — нулевым байтом.

Вывод результата обработки должен производиться в формате, соответствующем типу результата, т.е. как вещественного числа.

Указания:

- Выполнить программу в виде «вечного цикла» с возможностью его завершения вводом управляющего символа (например, Esc).
- Ввод символов с клавиатуры производить без автоматической выдачи эхо — печати на монитор, для чего воспользоваться функцией ввода символа `getch()`, подключив заголовочный файл **conio.h** Эхо — печать следует предусмотреть только для информационных символов.
- Для фильтрации ASCII-символов воспользоваться операциями отношения. В тексте программы вместо численного значения ASCII-кодов символов использовать их представление в виде символьных констант.

ЛАБОРАТОРНАЯ РАБОТА № 3
ИНТЕРФЕЙС ВВОДА МАТРИЦЫ

Цель работы. Получение навыков программирования интерфейса ввода с разделением управляющих и информационных символов.

Тематика изучаемого материала: операторы управления и логические выражения.

Задание к лабораторной работе

Составить программу, осуществляющую ввод двумерной числовой матрицы с клавиатуры с расположением элементов на экране в виде прямоугольной таблицы, состоящей из **n** строк и **m** столбцов (см.рис.1.) Для ввода числовых значений элементов матрицы следует воспользоваться функцией из лабораторной работы №2

При этом размеры матрицы — числа **n** и **m** — задаются, исходя из размеров фактически введенной таблицы. Число столбцов в матрице **m** определяется фактическим числом элементов введенных в *первую строку матрицы*, а число строк **n** — по количеству фактически введенных строк.

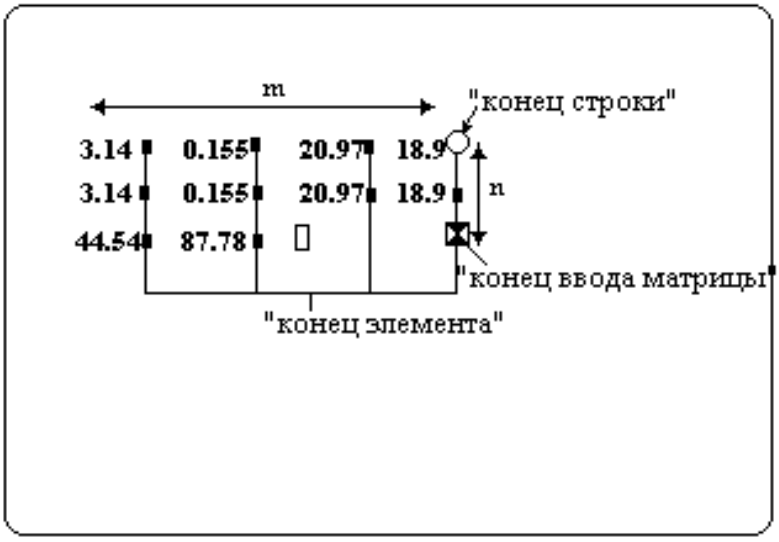


Рис. 1 — Вид поля ввода матрицы и расположение точек ввода управляющих символов

Программа должна работать в режиме ожидания событий — ввода управляющих символов с клавиатуры и обрабатывать следующие события:

1) ввод управляющего символа «**конец элемента**», в данном случае-последовательности информационных символов, образующих внешнее представление вещественного числа.

2) ввод управляющего символа «**конец строки**»,

3) ввод управляющего символа **«конец ввода матрицы»,**

4) ввод управляющего символа **«выход из программы**».

Обработку этих событий следует организовать по следующей логической схеме.

После ввода первой строки необходимо зафиксировать размер **m** матрицы.

При вводе второй и последующих строк, когда размер **m** уже зафиксирован, нужно блокировать реакцию на событие **«конец строки»** вплоть до заполнения строки требуемым количеством элементов.

На попытку ввода в последующие строки большего числа элементов, чем в первой строке требуется предусмотреть переход к вводу элемента следующей строки. Этот переход должен визуально отображаться перемещением курсора соответствующую позицию в поле ввода.

Значения вводимых элементов должны накапливаться в массиве вещественных чисел, и по завершении ввода всей матрицы (после ввода символа **«конец ввода матрицы»**), содержимое массива необходимо вывести на экран по формату вещественных чисел в виде матрицы. Этот контрольный вывод нужно воспроизвести на экране где — либо за пределами поля ввода, чтобы можно было сравнить вводимую матрицу с матрицей, образовавшейся в памяти. При правильной работе программы эти матрицы должны совпасть.

Указание. Позиционирование курсора на экране при вводе и выводе обеспечить с помощью функции **`void`** `gotoxy` **`(int`** `x,` **`int`** `y),` прототип которой дан в **conio.h** , при этом необходимо учесть, что эта функция работает с координатами символьного экрана.

ЛАБОРАТОРНАЯ РАБОТА № 4
ОБЪЕДИНЕНИЯ, БИТОВЫЕ ПОЛЯ И ПОРАЗРЯДНЫЕ ОПЕРАЦИИ

Цель работы получение навыков программирования поразрядных операций на C++, работы с объединениями и битовыми полями.

Тематика изучаемого материала: объединения и битовые поля, выражения с поразрядными операциями.

Задание к лабораторной работе

I. Определить в программе тип структуры с битовыми полями **struct** BY, общим размером 32 бита (4 байта), размеры битовых полей выбрать таким образом, чтобы одно из них по положению совпало с расположением двоичного порядка во внутреннем представлении числа типа **float**. Формат внутреннего представления числа типа float приведен в **Приложении В**.

Назначая размеры элементов структуры с битовыми полями, необходимо учесть два фактора:

a) размер элемента не может превышать принятого в данной системе размера int или, что эквивалентно, unsigned;

b) элементы объекта типа структура — битовое поле располагаются в адресном пространстве по порядку их записи в описании структуры, *начиная с младших битов*.

II. Сформировать в программе три объекта типа **union** каждый с двумя полями одинаковой длины разного типа:

```
union {
long L;
unsigned long UL;
} LUL;

union {
float F;
unsigned long UL;
} FUL;

union {
```

```
float F;
BY Bit
} FBY;
```

III. Составить 2 подпрограммы — функции, принимающие по два аргумента:

a) для функции 1 первый типа **long,** второй — типа **char***.

b) для функции 2 первый типа l **float,** второй — типа **char***.

В результате работы функций в массив, передаваемый вторым аргументом, помещается последовательность из 32 символов **'0'** и **'1'**, отражающую состояние битов в первом аргументе. Для удобства использования можно сделать эти функции, возвращающими указатель на символьный массив второго аргумента, после его заполнения внутри функций.

Указание. Внутри функции 1следует использовать объект LUL, внутри функции 2 — объект FUL, в обеих функциях поразрядные операции сдвига кода и поразрядную операцию «И»

Продемонстрируем на примере, как эта функция должна работать. Пусть эта функции имеет интерфейсы вида:

```
 char* LtoBit(long T, char* buf)

.char* FtoBit(float T, char* buf)
```

тогда при выполнении фрагмента программы

```
char bf[33];
cout<< LtoBit(1, bf)<<endl;
cout<< FtoBit(1, bf)<<endl;
```

в выходной поток должно быть выведено:

00000000000000000000000000000001

00111111100000000000000000000000

Строки отражают состояния битов в 32-х разрядном слове, где содержится двоичный код 1 в формате типов **long** и **float** соответственно

IV.Составить программы двух функций осуществляющих вывод значений *порядка* и *мантиссы* числа типа **float.**

IV.Составить программу осуществляющую

Ввод вещественных чисел с клавиатуры в переменную типа **float.**

Присвоение переменной типа **long** значение, хранимое в переменной типа **float**

Вывод на экран в битовом представлении введенных значений сохраненных как **long** и как **float,** воспользовавшись, соответственно, функциями `LtoBit` и `FtoBit`. В результате на экране должны появиться последовательности нулей и единиц соответствующие состоянию бит в кодах введенного числа.

Для вещественных типов вывести значения *мантиссы* и *порядка* числа во внутреннем представлении.

Указания

- Для реализации программы следует использовать объект типа **union** (объединение), и поразрядные операции языка С++.
- Учесть, что поразрядные операции в С++ применимы только к операндам целого типа (**int, unsigned** и т.п.).
- Получение значений порядка и мантиссы числа с плавающей точкой следует выполнить с использованием *битовых полей*.

ЛАБОРАТОРНАЯ РАБОТА № 5
ПРОГРАММИРОВАНИЕ ОДНОСВЯЗНОГО СПИСКА

Цель работы. Получение навыков работы с указателями и динамическим распределением памяти.

Задание к лабораторной работе

Составить программу, реализующую универсальный *контейнер* в форме *односвязного списка*.

Определение1: *Контейнером* называют программный объект, способный вмещать в себя другие объекты.

Определение 2: *Односвязным списком* называется схема организации множества элементов — данных, при которой с каждым элементом связан вспомогательный атрибут — ссылка на следующий элемент (см. рис. 2.)

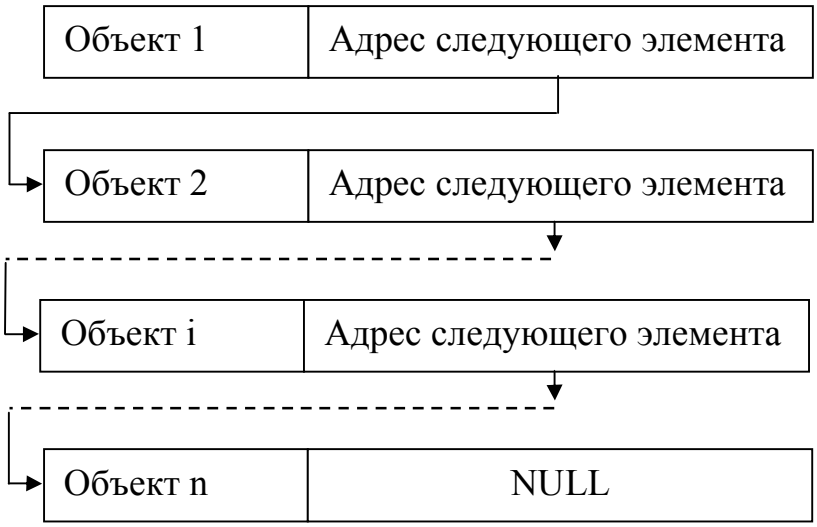


Рис. 2 — Структура односвязного списка

Необходимо реализовать следующие функции:

- дополнение контейнера объектами (теоретически в неограниченном количестве);
- удаление объектов из контейнера;
- подсчет количества элементов в контейнере;
- доступ к элементам по их номеру (индексу).

При программировании следует использовать указатели C++ и операторы динамического распределения памяти **new** и **delete**.

Реализовать контейнер для хранения объектов типа **int**;

В качестве усложненного варианта предлагается создать контейнер для хранения строк произвольной длины.

ЛАБОРАТОРНАЯ РАБОТА № 6
ПОСТРОЕНИЕ КЛАССА «ТРЕХМЕРНЫЙ ВЕКТОР»

Цель работы. Практическое ознакомление с правилами составления протоколов описаний классов C++, получение навыков составления элементарных программ с типами данных «объект — экземпляр класса».

Тематика изучаемого материала. Протоколы классов. Функции — элементы классов. Регламент доступа. Конструкторы и деструкторы.

Задание к лабораторной работе

Составить описание класса для объектов — векторов в трехмерном пространстве, считая, что компоненты векторов представлены вещественными числами типа **double**. Компоненты векторов должны быть скрыты (инкапсулированы) в объекте. Предусмотреть применение двух конструкторов

а) для инициализации векторов нулевыми компонентами и

б) заданным набором компонент.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «*Конструктор 1*», «*Деструктор*».

I. С помощью *функций — элементов* класса обеспечить

1) доступ к элементам вектора (чтение/запись);
2) вычисление модуля вектора;
3) копирование вектора;
4) умножение вектора на скаляр
5) нормировку вектор (получение вектора единичной длины).

II. С помощью *внешних функций* обеспечить двуместные операции над векторами A и B:

а) с помощью *функций — операторов*:

1) сложение (C = A + B);

2) вычитание (C = A – B);

3) векторное произведение (C = A * B);

б) с помощью *дружественных функций:*

1) скалярного произведения двух векторов;
2) косинуса угла между двумя векторами, полагая угол лежащем в пределах [0^o,180^o]);
3) величины угла в градусах между векторами в пределах [0,180].

Функции доступа к элементам и функцию, вычисляющую длину вектора определить внутри протокола класса, а остальные функции — элементы определить вне протокола класса.

Переопределить оператор << для вывода на экран компонент вектора в удобной форме, например, в виде строки:

`x =<значение x>  y =<значение y> z =<значение z>`

Предусмотреть в конструкторах и деструкторе вывод информационных сообщений и исследовать, в каких местах программы происходит автоматический вызов конструкторов и деструктора. Объяснить, почему так происходит.

Математические сведения, необходимые для программирования методов векторной алгебры представлены в ***Приложении А***.

Указание:Для расчета угла воспользуйтесь функцией вычисления арккосинуса `acos (double y, double x)` подключив заголовочный файл **math.h.**

ЛАБОРАТОРНАЯ РАБОТА № 7
ПОСТРОЕНИЕ КЛАССА ДЛЯ ПРЕДСТАВЛЕНИЯ МАТЕМАТИЧЕСКОГО ОБЪЕКТА «МАТРИЦА»

Цель работы. Освоение методов использования динамической памяти, изучение свойства полиморфизма, реализуемого перегрузкой функций и операций в классах C++.

Тематика изучаемого материала. Использование динамической памяти в классах. Роль конструкторов и деструктора. Конструктор копирования. Переопределение операции присваивания.

Задание к лабораторной работе

Составить описание класса для объектов — прямоугольных матриц, задаваемых массивом вещественных числами типа **double**, располагающегося в памяти по строкам. Компоненты матрицы должны быть скрыты (инкапсулированы) в объекте.

Предусмотреть применение конструкторов:

а) для инициализации квадратной матрицы заданного размера с заданными компонентами;

б) для инициализации прямоугольной матрицы заданных размеров с заданными компонентами.

Все конструкторы должны создавать объекты в динамической памяти (оператор **new**), а деструктор — освобождать память оператором **delete**.

Организовать в конструкторах и деструкторе вывод на экран информационных сообщений, например, «Конструктор 1», «Деструктор».

С помощью функций и операторов — элементов класса обеспечить:

– доступ к элементам матрицы по индексу строки и столбца (чтение/запись);

– проверку возможности умножения двух матриц;

– проверку возможности сложения двух матриц;

– переопределите операторы +=,-= и *= для объектов-матриц.

С помощью внешних функций обеспечить двуместные операции над матрицами A и B с получением новой матрицы C:

– сложение (C = A + B);

– вычитание (C = A – B);

– произведение (C = A * B).

Выполнению этих операций должна предшествовать проверка возможности их выполнения над данными объектами.

Используйте перегрузку функции — оператора *= для умножения матрицы на скаляр.

Организовать вычисление следующих скалярных величин:

– следа матрицы — суммы элементов на главной диагонали (для квадратных матриц);

– максимального элемента матрицы;

– минимального элемента матрицы;

Создайте функцию вывода на экран матрицы в построчной форме.

Для выравнивания позиций воспользуйтесь функцией-манипулятором потока setw(n).

При этом функции доступа к элементам определить внутри протокола класса, а остальные функции — элементы определить вне протокола класса.

ЛАБОРАТОРНАЯ РАБОТА № 8
ИЗУЧЕНИЕ МЕХАНИЗМА ОДИНОЧНОГО НАСЛЕДОВАНИЯ КЛАССОВ

Цель работы. Изучение свойств одиночного наследования в классах C++.

Тематика изучаемого материала. Одиночное наследование классов C++, взаимодействие конструкторов базового и производного классов, регламент доступа в производном классе к элементам своего базового класса. Виртуальные функции и позднее связывание.

Задание к лабораторной работе

1. Составить описание иерархической системы классов, состоящей из трех компонент:

   1) базового класса (верхний уровень иерархии),

   2) класса производного от базового (второй уровень иерархии)

   3) класса производного от производного(третий уровень иерархии).

При организации иерархической системы использовать public-наследование.

В каждом из классов создать блоки с различным уровнем доступа

private
protected
public

Поля данных в класса выполнить в форме символьных строк (**char***), память под которые выделяется динамически при создании объектов. Для удобства работы рекомендуется использовать строки с семантическим содержанием, отражающим расположение данных, например:

«*Приватные данные базового класса*» и т.д.

Предусмотреть применение конструкторов

а) конструктора без параметров, заполняющего поля объектов фиксированными строками;

б) конструктора, инициализирующего поля объекта с заданными строками.

Решить вопрос о необходимости деструкторов и, если они необходимы, создать их. Снабдить конструкторы и деструкторы выводом на экран информационных сообщений, например,

```
«Конструктор базового класса» и т.д.
```

С помощью функций — элементов класса обеспечить доступ к полям объектов.

2. Составить исполняемую main — программу, создающую экземпляры объектов, организовать вывод на экран содержимого полей объектов, используя соответствующие методы доступа. В производных классах вывести содержимое унаследованных полей.

3. Изменить модификатор наследования (**`public`** на **`private`**), запустить программу и прокомментировать результат. Организовать вывод унаследованных полей в этом случае.

4. Провести исследования и ответить на вопрос: как изменяется тип доступа наследуемых полей в зависимости от модификатора наследования

5. Создать в каждом из классов функцию-элемент с одинаковой сигнатурой. Проследить, как происходит ее перегрузка при наследовании.

6. Функцию, введенную в предыдущем разделе, объявить в базовом классе как виртуальную. Отметить изменения в характере ее перегрузки при наследовании. Сделать выводы.

7. Сделать базовый класс абстрактным, запустить программу и дать комментарии к результату.

8. Реализовать механизм позднего связывания, используя обращение к объекту производного класса через указатель на объект базового класса.

Контрольные вопросы к лабораторной работе № 8

1. В чем различие private — наследования, protected — наследования и public — наследования?

2. К какому типу наследования приводит реализация следующего кода: class A ...; class B : A ...;

3. Что такое виртуальные функции и для чего они используются?

4. Какие классы называются абстрактными? Каково их назначение?

5. При каких обстоятельствах в конструкторе производного класса следует обязательно указывать конструктор базового класса?

6. Каким образом можно отказаться от наследования части протокола базового класса?

ЛАБОРАТОРНАЯ РАБОТА № 9
СОЗДАНИЕ КЛАССА ДЛЯ ПРЕДСТАВЛЕНИЯ ОБЪЕКТА «ДАТА/ВРЕМЯ»

Цель работы. Создание класса для представления в едином объекте даты и времени с возможностью осуществления операций сложения и вычитания для связи интервалов времени и дат.

Тематика изучаемого материала. Перегрузка функций и операторов внутри класса.

Задание к лабораторной работе

Для хранения даты и времени создать в закрытой области класса переменную типа **double**, выражающей время в сутках, прошедшее от заданной начальной точки отсчета до заданного момента времени.

Предусмотреть возможность инициализации объекта текущей датой и временем с системных часов компьютера, для этого следует воспользоваться функциями

```
void getdate( struct date*   datep );//запрос даты

void gettime( struct time *  timep ); //запрос времени
```

Прототипы этих функций находятся в заголовочном файле **dos.h.**

Результат работы этих функций возвращается через переданные по указателю структуры `date` и `time`, определенные в том же заголовочном файле, что и функции — **dos.h**. Определения этих структур выглядят таким образом:

```
struct  date    {
  int     da_year;   // год - 2006
  char    da_day;    // день месяца
  char    da_mon;    // месяц (1- январь)
};
struct  time{
  unsigned char    ti_min;   // минуты
  unsigned char    ti_hour;  // часы
```

```
  unsigned char    ti_hund;   // сотые доли секунды
  unsigned char    ti_sec;    // секунды
};
```

Построить класс для описания объекта, хранящего дату и время суток в едином объекте.

Перегрузка операций сложения и вычитания. Реализация методов «день недели» и «текущая дата и время».

ЛАБОРАТОРНАЯ РАБОТА № 10
ПОСТРОЕНИЕ КЛАССА ДЛЯ ПРЕДСТАВЛЕНИЯ МАТЕМАТИЧЕСКОГО ОБЪЕКТА «ОБЫКНОВЕННАЯ ДРОБЬ»

Цель работы. Создание классов математических объектов с перегрузкой операций.

Задание к лабораторной работе

1. Составить описание класса для представления типа «обыкновенная дробь» вида:

$$C = \frac{A}{B},$$

где A и B — целые числа.

2. Перегрузить операции 4-х арифметических действий (+, – , * , /) для дробей.

3. Предусмотреть операцию «сокращение дроби» — удаления общих множителей числителя и знаменателя.

4. Обеспечить операцию декомпозиции неправильной дроби на целую часть и правильную дробь, используя операцию деления с остатком.

ЛАБОРАТОРНАЯ РАБОТА № 11
ПОСТРОЕНИЕ КЛАССА ДЛЯ ПРЕДСТАВЛЕНИЯ МАТЕМАТИЧЕСКОГО ОБЪЕКТА «ПОЛИНОМ»

Цель работы. Создание классов математических объектов с перегрузкой операций.

Тематика изучаемого материала. Перегрузка операций, использование динамической памяти.

Задание к лабораторной работе

1. Составить описание класса для представления типа «**полином**» вида:

$$\mathbf{P(x)=a_0 + a_1x + a_2x^2 + ...+a_nx^n,}$$

где a_i — вещественные коэффициенты.

2. Обеспечить выполнение 3-х арифметических действий над этими объектами (+ , – , *).

3. Предусмотреть операцию деления полиномов с остатком, используя «деление углом» (алгоритм Эвклида). Пример применения алгоритма Эвклида к полиномам дан в **Приложении Г**.

4. Обеспечить операцию декомпозиции неправильной дроби на целую часть и правильную дробь, используя операцию деления с остатком.

5. Для всех типов организовать перегруженный оператор присваивания.

Указание: все арифметические операторы реализовать путем перегрузки операций с использованием свойства дружественности.

ЛАБОРАТОРНАЯ РАБОТА № 12
ПОСТРОЕНИЕ ШАБЛОННОГО КЛАССА ДЛЯ ПРЕДСТАВЛЕНИЯ МАТЕМАТИЧЕСКОГО ОБЪЕКТА «ПОЛИНОМИАЛЬНАЯ ДРОБЬ»

Цель работы. Изучение механизма применения шаблонов функций как средства агрегирования программных модулей.

Тематика изучаемого материала. Перегрузка операций, использование динамической памяти.

Задание к лабораторной работе

Данная лабораторная работа базируется на результатах лабораторной работ № 10 и № 11.

1. Используя исходный коды модуля «обыкновенная дробь», тип «полином» и метод шаблонов составить описание класса для представления объектного типа «полиномиальная дробь»:

$$W(x)= \frac{b_0 + b_1x + b_2x^2 + ...+b_mx^m}{a_0 + a_1x + a_2x^2 + ...+a_nx^n},$$

где a_i и b_i — вещественные числа;

m и n — целые числа.

2. Обеспечить выполнение 3-х арифметических действий над этими объектами (+ , – , *).

3. Обеспечить операцию декомпозиции неправильной дроби на целую часть и правильную дробь, используя операцию деления с остатком. (Здесь целая часть — полином, остаток — правильная полиномиальная дробь, у которой степень полинома числителя *меньше*, чем степень полинома знаменателя)

4. На основе кода программного модуля «обыкновенная дробь» создать *шаблон* с параметризацией типа объектов в числителе и знаменателе дроби;

5. Сформировать классы реализации «полиномиальная дробь», конкретизируя параметр созданного шаблона типом «полином» и типом **`int`**. В первом случае шаблонный класс должен корректно работать с полиномиальными дробями, во втором — обыкновенными.

ПРИЛОЖЕНИЕ А

Сведения из векторной алгебры

Пусть A, B и C — вектора в трехмерном пространстве с компонентами { Ax , Ay , Az }, { Bx , By, Bz } и { Cx , Cy, Cz} соответственно. Тогда для C = A + B имеет место:

$$Cx = Ax + Bx;$$
$$Cy = Ay + By;$$
$$Cz = Az + Bz.$$

Модулем вектора A называют число m = | A | , определяемое как корень квадратный из суммы квадратов компонент, т.е.

$$m = \sqrt{Ax^2 + Ay^2 + Az^2}\ .$$

Векторным произведением двух векторов A и B называют вектор C= [AB], компоненты которого определяются на основе следующих соотношений:

$$Cx = Ay*Bz - Az*By;$$
$$Cy = Az*Bx - Ax*Bz;$$
$$Cz = Ax*By - Ay*Bx;$$

Вектор C перпендикулярен векторам A и B одновременно, его направление совпадает с движением правого винта, вращаемого от A к B. Для модуля вектора C справедливо соотношение:

$$| C | = |A| *|B| *|\mathrm{Sin}\ a|,$$ где a — угол между векторами.

Скалярным произведением двух векторов A и B называют скалярную величину s = (AB), определяемую как

$$s = Ax*Bx+Ay*By+Az*Bz\ .$$

Для скалярного произведения имеет место соотношение:

$$s = |A| *|B| *\mathrm{Cos}\ a,$$

где a — угол между векторами

ПРИЛОЖЕНИЕ Б

1. Кодировка некоторых символов ASCII (DOS)

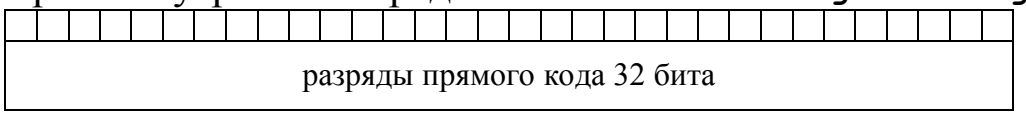
32	пробел	64	@	96	`	128	А	160	а
33	!	65	A	97	a	129	Б	161	б
34	"	66	B	98	b	130	В	162	в
35	#	67	C	99	c	131	Г	163	г
36	$	68	D	100	d	132	Д	164	д
37	%	69	E	101	e	133	Е	165	е
38	&	70	F	102	f	134	Ж	166	ж
39	'	71	G	103	g	135	З	167	з
40	(	72	H	104	h	136	И	168	и
41	)	73	I	105	i	137	Й	169	й
42	*	74	J	106	j	138	К	170	к
43	+	75	K	107	k	139	Л	171	л
44	,	76	L	108	l	140	М	172	м
45	-	77	M	109	m	141	Н	173	н
46	.	78	N	110	n	142	О	174	о
47	/	79	O	111	o	143	П	175	п
48	0	80	P	112	p	144	Р	224	р
49	1	81	Q	113	q	145	С	225	с
50	2	82	R	114	r	146	Т	226	т
51	3	83	S	115	s	147	У	227	у
52	4	84	T	116	t	148	Ф	228	ф
53	5	85	U	117	u	149	Х	229	х
54	6	86	V	118	v	150	Ц	230	ц
55	7	87	W	119	w	151	Ч	231	ч
56	8	88	X	120	x	152	Ш	232	ш
57	9	89	Y	121	y	153	Щ	233	щ
58	:	90	Z	122	z	154	Ъ	234	ъ
59	;	91	[	123	{	155	Ы	235	ы
60	<	92	\	124	|	156	Ь	236	ь
61	=	93	]	125	}	157	Э	237	э
62	>	94	^	126	~	158	Ю	238	ю
63	?	95	_	127	нет	159	Я	239	я

2. Коды некоторых функциональных клавиш

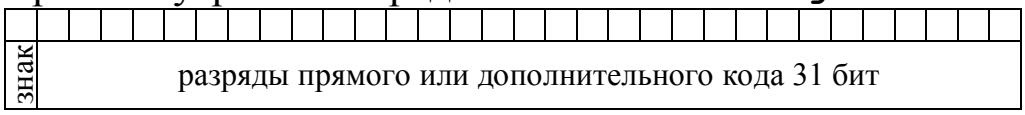
Клавиша	Код
Tab	9
backspace	8
Enter	13
Esc	27

ПРИЛОЖЕНИЕ В

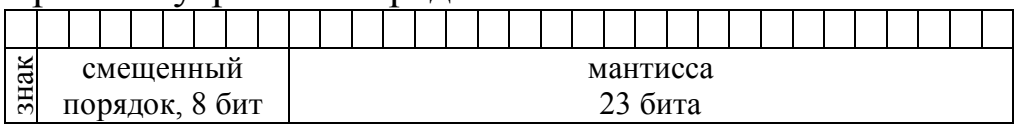
Формат внутреннего представления типа **`unsigned long`**:



Формат внутреннего представления типа **`long`**

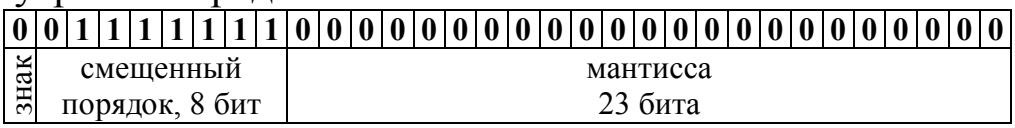


Формат внутреннего представления типа **`float`**

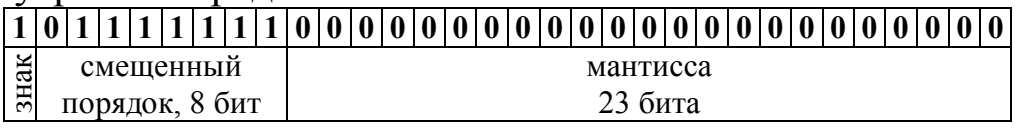


Примеры внутреннего представления чисел 1 и –1:

Внутреннее представление числа 1 типа **`float`**

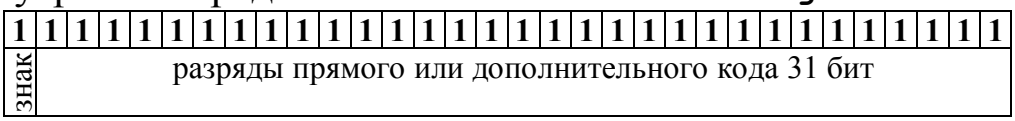


Внутреннее представление числа –1 типа **`float`**



Внутреннее представление числа 1 типа **`long`**

Внутреннее представление числа –1 типа **`long`**



ПРИЛОЖЕНИЕ Г

Пример деления полинома по алгоритму Эвклида.

Пусть даны два полинома:
полином делимое: $X^4 + 3X^3 + 5X^2 + 4X + 5$
и полином делитель: $X^2 + 8X + 7$

Алгоритм Эвклида — это правило деления чисел «в столбик».

Запишем этот алгоритм деления в традиционной форме:

X^4	$+3X^3$	$+5X^2$	$+4X$	$+5$	X^2	$+8X$	$+7$
$-X^4$	$-8X^3$	$-7X^2$			X^2	$-5X$	$+38$
	$-5X^3$	$-2X^2$	$+4X$				
	$5X^3$	$+40X^2$	$+35X$				
		$38X^2$	$+39X$	$+5$			
		$-38X^2$	$-304X$	-266			
			$-265X$	-261			

В результате получили:

$$\frac{X^4 + 3X^3 + 5X^2 + 4X + 5}{X^2 + 8X + 7} = X^2 - 5X + 38 - \frac{265X + 261}{X^2 + 8X + 7}$$

Таким образом, при делении полиномов получается полином и «правильная полиномиальная дробь», порядок полинома в числителе которой меньше порядка полинома в знаменателе.

Если проводить аналогию между делением целых чисел с остатком, то полином $X^2 - 5X + 38$ следует рассматривать как целую часть от деления, а полином — $265X - 261$ как остаток от деления нацело.

ЛИТЕРАТУРА

1. Франка П. C++: учебный курс: Пер. с англ. П. Бибикова. — СПб.: Питер, 2005. — 521 с

2. Павловская Т.А., Щупак Ю.А. C++. Объектно-ориентированное программирование: практикум: учебное пособие для вузов. — СПб.: Питер, 2005. — 464 с

3. Павловская Т.А. C/C++: Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2002. — 460 с

4. Боровский А.Н. Самоучитель C++ и Borland C++ Builder: самоучитель. — СПб.: Питер, 2005. — 255 с.