МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»
Руководитель образовательной программы

_____ И.Л. Артемьева
«21» 08 2015 г.

«УТВЕРЖДАЮ»
Заведующая кафедрой прикладной математики, механики, управления и программного обеспечения

_____ И.Л. Артемьева
«21» 08 2015

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Объектно-ориентированное программирование

Направление подготовки - 010500.62 «Математическое обеспечение и администрирование информационных систем подготовки»
профиль «Технология программирования»
Форма подготовки (очная)

курс 2 семестр 4
лекции ___ час.
практические занятия _____час.
лабораторные работы 72час.
в том числе с использованием МАО лек._____/пр._____/лаб 72 час.
в том числе в электронной форме лек._____/пр._____/лаб 36 час.
всего часов аудиторной нагрузки 72 час.
в том числе с использованием МАО 72 час.
самостоятельная работа 72 час.
в том числе на подготовку к экзамену _____ час.
курсовая работа / курсовой проект не предусмотрены
зачет 4 семестр
экзамен не предусмотрен

Рабочая программа составлена в соответствии с требованиями федерального государственного образовательного стандарта высшего образования, утвержденного приказом Министерства образования и науки РФ от 12 марта 2015 г. № 222

Рабочая программа обсуждена на заседании кафедры прикладной математики, механики, управления и программного обеспечения, протокол № 7 от « 4 » июля 2015г.

Заведующая кафедрой прикладной математики, механики, управления и программного обеспечения Артемьева И.Л., д.т.н., профессор

Составители: Заведующая кафедрой прикладной математики, механики, управления и программного обеспечения Артемьева И.Л., д.т.н., профессор,
доцент кафедры прикладной математики, механики, управления и программного обеспечения Остроухова С.Н., к.т.н.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

ABSTRACT

Bachelor's/Specialist's/Master's degree in 02.03.03 – Software and Administration of Information Systems

Study profile/ Specialization/ Master's Program "Title" Programming technology

Course title: The development of object-oriented applications

Basic part of Block 1, _4_credits

Instructor: Artemeva I., Ostroukhova S.

At the beginning of the course a student should be able to use main methods and principles of solving problems with the help of computers using a high-level programming language

Learning outcomes: ability to use knowledge of main conceptual provisions of functional, logic, object-oriented and visual trends of programming, methods and tools of the development of programs in the framework of these trends; ability to use knowledge of the methods of software design and production, construction principles, structure and the methods of work with instrumental tools used for creating software; readiness to use the skills of choice, design, implementation, quality assessment and efficiency analysis of software to solve problems from various subject spheres; readiness to use main models of information technologies and the methods of its application to solve problems in subject spheres.

Course description: the methods of the creation of object-oriented programs in C++.

Main course literature:

1. Pavlovskaya T.A. Shchupak Yu.A. C/C++. Strukturnoe i ob"ektno-orientirovannoe programmirovanie: praktikum [Structural and object-oriented programming: practical work]. Saint Petersburg, Piter, 2010. 347 p.

http://lib.dvfu.ru:8080/lib/item?id=chamo:418970&theme=FEFU

2. Stroustrup B. Yazyk programmirovaniya C++. Spetsial'noye izdanie [The C++ programming language. Special edition]. Moscow, BINOM, 2011. 1135 p.

https://lib.dvfu.ru:8443/lib/item?id=chamo:418460&theme=FEFU

3. Pavlovskaya T.A. C/C++. Programmirovanie na yazyke vysokogo urovnya [C/C++. Programming on a high-level language]. Saint Petersburg, Piter, 2011. 460 p.

https://lib.dvfu.ru:8443/lib/item?id=chamo:307692&theme=FEFU

4. Golova S.Yu., Nemtsova T.I., Terentev A.I. Programmirovanie na yazyke C++ [Programming in C++]. Moscow, Infra-M, 2012. 512 p.

http://znanium.com/catalog.php?bookinfo=244875

Form of final control: pass-fail exam.

АННОТАЦИЯ

Рабочая программа дисциплины «Объектно-ориентированное программирование» разработана для студентов 2 курса, обучающихся по направлению 02.03.03 «Математическое обеспечение и администрирование информационных систем», профиль «Технология программирования».

Трудоемкость дисциплины 4 зачетных единицы (144 часа). Дисциплина реализуется в 4 семестре. В 4 семестре дисциплина содержит 0 часов лекций, 0 часов практических занятий, 72 часа лабораторных работ, из них 0 часов лекций, 0 часов практических занятий, 72 часов лабораторных работ с использованием методов активного обучения. На самостоятельную работу студента отводится 72 часа.

Дисциплина «Объектно-ориентированное программирование» базируется на дисциплинах «Математические основы информатики и программирования», “Основы алгоритмизации”, «Компьютерный практикум» и "Практикум по программированию». Знания, полученные при ее изучении, будут использованы в дисциплинах учебного плана, связанных с использованием ЭВМ, а также в практической деятельности бакалавра при разработке программных систем.

Цель дисциплины - усвоение и закрепление методов создания объектно-ориентированных программ на C++, знакомство с понятиями абстрактного класса, шаблонами классов.

Задачи:

1. Изучить основные концептуальные положения объектно-ориентированного программирования, а также механизмы, методы и средства разработки приложений в рамках данного направления
2. Изучить язык программирования C++, научиться грамотно его использовать.
3. Научиться использовать методы разработки объектно-ориентированных программ.

В результате изучения данной дисциплины у обучающихся формируются следующие общепрофессиональные компетенции (элементы компетенций).

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК7 Способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений	Знает	основные положения и концепции объектно-ориентированного программирования
	Умеет	использовать методы объектно-ориентированного программирования при создании программных систем
	Владеет	методами, способами и программными средствами для разработки объектно-ориентированных программ
ОПК8 Способность использовать знания методов проектирования и производства программного продукта, принципов построения, структуры и приемов работы с инструментальными средствами, поддерживающими создание программного обеспечения	Знает	Методы проектирования объектно-ориентированных приложений
	Умеет	Проектировать требуемый набор классов и методов работы с ними при создании объектно-ориентированных приложений
	Владеет	Навыками разработки объектно-ориентированных программных средств по проекту
ОПК11 Готовность использовать навыки выбора, проектирования, реализации, оценки качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях	Знает	Методы оценки качества проекта объектно-ориентированных приложений
	Умеет	Проектировать требуемый набор тестов для оценки качества объектно-ориентированных приложений
	Владеет	Навыками создания пакета тестов для оценки качества объектно-ориентированных приложений
ПК2 готовность к использованию основных моделей информационных технологий и способов их применения для решения задач в предметных областях	Знает	принципы наследования, инкапсуляции и полиморфизма, положенные в разработку объектно-ориентированных языков
	Умеет	использовать принципы наследования, инкапсуляции и полиморфизма при создании объектно-ориентированных приложений
	Владеет	методами проектирования и разработки программ, используя полиморфные функции и операции, разные типы наследования и создавая методы доступа к объектам разных классов (инкапсуляция)

Для формирования вышеуказанных компетенций в рамках дисциплины «Объектно-ориентированное программирование» применяются следующие методы активного/ интерактивного обучения: метод проектов.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Лекции не предусмотрены учебным планом.

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (72 час.)

Лабораторная работа 1. Объекты и классы C++. (12 часов)

Получение базовых знаний в области объектно-ориентированного программирования, изучение основных понятий и парадигм: определение класса для заданной задачи, программирование методов класса, типы конструкторов, прототипы функций.

Лабораторная работа 2. Работа с классами в C++. (10 часов).

Получение практических навыков в области перегрузки операторов.

Разработка классов для работы с данными разных типов: стек, список, вектор, комплексные числа и т.д. Перегрузка операций. Дружественные функции.

Лабораторная работа 3. Классы и динамическое выделение памяти. (12 часов)

Получение практических навыков в выделении динамической памяти под объекты классов.

Лабораторная работа 4. Наследование. (18 часов)

Получение практических навыков в области одиночного и множественного наследования классов. Базовый класс. Виртуальны методы.

Лабораторная работа 5. Дружественные классы. (12 часов)

Получение практических навыков работы в области разработки дружественных классов

Лабораторная работа 6. Потоковый ввод и вывод. Iostream. (8 часов)

Получение практических навыков работы с потоками ввода и вывода C++.

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Объектно-ориентированное программирование» представлено в Приложении 1 и включает в себя: план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию; характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению; требования к представлению и оформлению результатов самостоятельной работы; критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

№ п/п	Контролируемые разделы/темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства - наименование	
				текущий контроль	промежуточная аттестация
1	Объекты и классы C++.	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 1	Зачет
2	Работа с классами в C++.	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 2	Зачет
3	Классы и динамическое выделение памяти	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 3	Зачет
4	Наследование	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 4	Зачет
5	Дружественные классы	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 5	Зачет
6	Потоковый ввод и вывод. Iostream.	ОПК7 ОПК8 ОПК11	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа	Зачет

		ПК2		ПР-6 лабораторная работа 6	

Типовые контрольные задания, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

1. С/С++. Структурное и объектно-ориентированное программирование: практикум / Т.А. Павловская, Ю.А. Щупак.- СПб.: Питер, 2010.- 347 с. http://lib.dvfu.ru:8080/lib/item?id=chamo:418970&theme=FEFU
2. Язык программирования С++. Специальное издание: пер. с англ. / Бьерн Страуструп. - М.: Вильямс, 2011. — 1135с. https://lib.dvfu.ru:8443/lib/item?id=chamo:418460&theme=FEFU
3. С/С++. Программирование на языке высокого уровня: учебник для вузов для магистров и бакалавров / Т. А. Павловская. СПб.: Питер, 2011.-460 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:307692&theme=FEFU
4. Программирование на языке С++: Учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев; Под ред. Л.Г. Гагариной. - М.: ИД ФОРУМ: ИНФРА-М, 2012. - 512 с. http://znanium.com/catalog.php?bookinfo=244875
5. Дейл, Н. Программирование на С++ [Электронный ресурс] / Н. Дейл, Ч. Уимз, М. Хедингтон; Пер. с англ. - М.: ДМК Пресс, 2007. - 672 с. http://znanium.com/catalog.php?bookinfo=407353
6. Объектно-ориентированное программирование в С++: лекции и упражнения [Электронный ресурс]: Учебное пособие для вузов / Ашарина И.В. - М.: Горячая линия - Телеком, 2012. - http://www.studentlibrary.ru/book/ISBN9785991270014.html

Дополнительная литература

1. Дизайн и эволюция С++: [пер. с англ.] / Бьерн Страуструп. Москва: ДМК Пресс, Санкт-Петербург: Питер, 2007. - 444 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:297090&theme=FEFU

2. Павловская Т.А. С/С++. Программирование на языке высокого уровня- СПб.: Питер, 2006. - 461 с. http://lib.dvfu.ru:8080/lib/item?id=chamo:237589&theme=FEFU
3. Царев, Р. Ю. Программирование на языке Си [Электронный ресурс]: учеб. пособие / Р. Ю. Царев. – Красноярск: Сиб. федер. ун-т, 2014. – 108 с. http://znanium.com/catalog.php?bookinfo=510946
4. Керниган Б., Ритчи Д. Язык программирования С. Пер. с англ. — М.: Издательский дом "Вильямс", 2012. — 289 с. http://lib.dvfu.ru:8080/lib/item?id=chamo:666721&theme=FEFU
5. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб. пособие для вузов. М.: Финансы и статистика, 2009. - 600 с.. http://lib.dvfu.ru:8080/lib/item?id=chamo:355876&theme=FEFU
6. Хабибуллин И.Ш. Программирование на языке высокого уровня С/С++. - СПб.: БХВ-Петербург, 2006. - 499 с. http://znanium.com/catalog.php?bookinfo=356906
7. Скользкие места С++. Как избежать проблем при проектировании и компиляции ваших программ [Электронный ресурс] / Стефан К. Дьюхэрст. - М.: ДМК Пресс, 2006. http://www.studentlibrary.ru/book/ISBN5940740839.html
8. Холзнер С. Visual C++: Учебный курс.- СПб.: Питер, 2007. — 570 с: ил.
9. Прата Стивен. Язык программирования С++: Лекции и упражнения (5-е изд.), М.: ООО "И.Д. Вильямс", 2007. 1171 с.
10. Мэйерс, С. Эффективное использование С++. 55 верных способов улучшить структуру и код ваших программ [Электронный ресурс] / С. Мэйерс. - М.: ДМК Пресс, 2008. - 300 с. http://znanium.com/catalog.php?bookinfo=409179
11. Шилдт Г. Полный справочник по С++. М.: ИД "Вильямс", 4-е изд, 2006. - 791с.
12. Хортон А. Visual C++ 2010: полный курс .-Пер. с англ. - М. : ООО "И.Д. Вильямс", 2011. 1216 с. ISBN 978-5-8459-1698-3 (рус.)

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. http://window.edu.ru/resource/458/24458/files/metoopk.pdf Объектно-ориентированная методология разработки сложных систем: Учебное пособие / Т.В. Глотова. Пенза, 2001. – 49 с.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. Максимчук, М. Энгл, Б. Янг, Д. Коннален, К. Хьюстон. – М.: Издательский дом «Вильямс», 2008 – 720 с. [электронный

ресурс] .-Режим доступа:http://www.razym.ru/94003-gradi-buch-robert-a-maksimchuk-majkl-u-yengl.html
3. https://eknigi.org/programmirovanie/53044-obektno-orientirovannyj-analiz-i-proektirovanie.html Йордон Э., Аргила К.. Объектно-ориентированный анализ и проектирование систем. Изд.: Лори, 2007 – 284с. [электронный ресурс]
4. https://code-live.ru/tag/cpp-manual/ Портал о программировании
5. http://cppstudio.com/cat/274/ Язык программирования С++
6. http://www.c-cpp.ru/ Программирование на С и С++. Онлайн справочник программиста
7. http://www.proklondike.com/books/cpp.html Программирование
8. http://purecodecpp.com/ основы программирования на С++ для начинающих
9. http://progbook.ru/c/ Книги по С++
10. http://mexalib.com/view/2267 Компьютерная литература. Шилдт Г. Самоучитель С++ BHV - Санкт - Петербург, 2003. - 688 стр.
11. http://www.programmersclub.ru/main/ Клуб программистов
12. http://info-comp.ru/ Информационный портал. Все о компьютере и программировании для начинающих

Перечень информационных технологий и программного обеспечения

При осуществлении образовательного процесса студентами и профессорско-преподавательским составом используется следующее программное обеспечение:

1. Пакет прикладных программ Microsoft Office / Open Office.
2. Интегрированные среды разработки программ на языке С++.
3. Программное обеспечение электронного ресурса сайта ДВФУ, включая ЭБС ДВФУ.

При осуществлении образовательного процесса студентами и профессорско-преподавательским составом используются следующие информационно-справочные системы:

1. Научная электронная библиотека eLIBRARY.
2. Электронно-библиотечная система IPRbooks.
3. Доступ к электронному заказу книг в библиотеке ДВФУ, доступ к нормативным документам ДВФУ, расписанию, рассылке писем.

Лабораторные занятия проводятся в специализированном компьютерном классе.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Дисциплина «Объектно-ориентированное программирование» изучается в следующих организационных формах: лабораторное занятие; самостоятельное изучение теоретического материала; индивидуальные и групповые консультации.

Лабораторные работы

В результате выполнения лабораторных работ студент должен научиться формализовать поставленную задачу, составлять и оформлять программы на языке программирования C++, тестировать и отлаживать программы, использовать для разработки программ современные интегрированные среды разработки программ.

При выполнении лабораторной работы необходимо следовать методическим рекомендациям по ее выполнению. Результатом лабораторной работы является программа и отчет (документация), которые демонстрируется преподавателю в конце работы. Студент должен уметь отвечать на вопросы преподавателя, поясняя процесс создания программ и документа и выполнения работы.

Индивидуальные задания необходимо выполнять согласно рассмотренной технологии:

1. изучить словесную постановку задачи;
2. сформулировать математическую постановку задачи;
3. выбрать метод решения задачи, если это необходимо;
4. разработать схему алгоритма и спроектировать требуемые классы;
5. записать разработанный алгоритм на языке C++;
6. разработать контрольные тесты программы;
7. отладить программу;
8. написать отчет.

Содержание отчета

1. Титульный лист.
2. Неформальная постановка задачи (НПЗ).
3. Формальная постановка задачи (ФПЗ).
4. Проекты классов
5. Текст программы с комментариями.
6. Тестовые условия.

Работа с литературой

В процессе подготовки к лабораторным работам студентам необходимо обратить особое внимание на поиск и на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студентов свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем по каждой теме практического занятия, что позволяет студентам проявить свою индивидуальность в рамках выполнения индивидуального проекта, выявить широкий спектр мнений по изучаемой проблеме.

Самостоятельная работа студента

Основными формами самостоятельной работы студента являются:

- подготовка к лабораторным занятиям, зачету,
- изучение обязательной и дополнительной литературы,
- поиск информации по изучаемым темам в периодических изданиях и Интернете,
- изучение в рамках программы курса тем, не выносимых на лабораторные работы,
- оформление отчетов по лабораторным работам.

Контроль за выполнением работы студента производится в виде контроля каждого этапа работы (см. приложение 1).

Студент должен планировать график самостоятельной работы по дисциплине и придерживаться его.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Лекции проводятся с использованием проектора и внутренней системы портала ДВФУ. Лабораторные занятия проходят в аудиториях, оборудованных компьютерами типа Lenovo C360G-i34164G500UDK с лицензионными программами MicrosoftOffice 2013 и аудио-визуальными средствами проектор Panasonic DLPProjectorPT-D2110XE, плазма LG FLATRON M4716CCBAM4716CJ. Для выполнения самостоятельной работы студенты в жилых корпусах ДВФУ обеспечены Wi-Fi.

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Объектно-ориентированное программирование»

Направление подготовки – 02.03.03 «Математическое обеспечение и администрирование информационных систем»

профиль «Технология программирования»
Форма подготовки (очная)

Владивосток
2015

Самостоятельная работа состоит в выполнении на компьютере индивидуальных практических заданий по изучаемым темам (72 часа).

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Виды СРС	Всего часов	Форма контроля
1.	1-3 неделя обучения	Изучение теоретического материала к ЛР-1 по литературным источникам	4	Собеседование
		Оформление отчета по ЛР-1	8	Проверка отчета
2.	4-5 неделя обучения	Изучение теоретического материала к ЛР-2 по литературным источникам	4	Собеседование
		Оформление отчета по ЛР-2	8	Проверка отчета т
3.	6-8 неделя обучения	Изучение теоретического материала к ЛР-3 по литературным источникам	4	Собеседование
		Оформление отчета по ЛР-3	8	Проверка отчета
4.	9-12 неделя обучения	Изучение теоретического материала к ЛР-4 по литературным источникам	4	Собеседование
		Оформление отчета ЛР-4	8	Проверка отчета
5.	13-15 неделя обучения	Изучение теоретического материала к ЛР-5 по литературным источникам	4	Собеседование
		Оформление отчета ЛР-6	8	Проверка отчета
6.	16-18 неделя обучения	Изучение теоретического материала к ЛР-6 по литературным источникам	4	Собеседование
		Оформление отчета ЛР-6	8	Проверка отчета
Итого:			72	

Рекомендации по работе с литературой

Для более эффективного освоения и усвоения материала рекомендуется ознакомиться с теоретическим материалом по той или иной теме до проведения лабораторного занятия. Всю учебную литературу желательно изучать «под конспект».

Цель написания конспекта по дисциплине – сформировать навыки по поиску, отбору, анализу и формулированию учебного материала.

Работу с теоретическим материалом по теме можно проводить по следующей схеме:

- название темы;
- цели и задачи изучения темы;
- основные вопросы темы;

- характеристика основных понятий и определений, необходимых для усвоения данной темы;

- краткие выводы, ориентирующие на определенную совокупность сведений, основных идей, ключевых положений, систему доказательств, которые необходимо усвоить.

При работе над конспектом обязательно выявляются и отмечаются трудные для самостоятельного изучения вопросы, с которыми уместно обратиться к преподавателю при посещении консультаций, либо в индивидуальном порядке.

Подготовка к лабораторным работам

Подготовку к каждой лабораторной работе каждый студент должен начать с изучения теоретического материала и ознакомления с планом, который отражает содержание предложенной темы. Все новые понятия по изучаемой теме необходимо выучить наизусть и внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности студента свободно ответить на теоретические вопросы по теме задания, правильном выполнении лабораторной работы.

В процессе выполнения лабораторной работы студент должен создать требуемый документ (отчет) с помощью предлагаемого программного средства и выполнить требуемые в задании операции. Задание по лабораторной работе содержит методические указания по подготовке документа, который должен быть получен в результате выполнения работы. При подготовке к лабораторной работе следует их внимательно прочесть.

При формировании индивидуальных заданий по самостоятельной работе используется материал из работы: Программирование: метод. указания/ сост. Л.И.Прудникова - Владивосток: Изд-во ДВФУ, 2012. - 24 с.

Содержание отчета по лабораторной работе

1. Титульный лист.
2. Неформальная постановка задачи (НПЗ).
3. Формальная постановка задачи (ФПЗ).
4. Блок-схема.
5. Текст программы с комментариями.
6. Тестовые условия.

Критерии оценки отчетов по лабораторным работам (проектов)

– 100-86 баллов выставляется, если содержание и составляющие части соответствуют выданному заданию. Продемонстрировано владение навыками подготовки документа по теме. Фактических ошибок, связанных с пониманием структуры и содержания задания нет.

– 85-76 - баллов выставляется, если при выполнении задания допущено не более одной ошибки. Продемонстрировано владение навыками подготовки документа по теме. Фактических ошибок, связанных с пониманием структуры и содержания задания нет.

– 75-61 балл выставляется, если при выполнении задания допущено не более двух ошибок. Продемонстрировано навыки подготовки документа по теме. Допущено не более 2 ошибок, связанных с пониманием структуры и содержания задания.

60-50 баллов - если структура и содержание задания не соответствуют требуемым

Шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Приложение 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Объектно-ориентированное программирование»

Направление подготовки – 02.03.03 «Математическое обеспечение и администрирование информационных систем»

профиль «Технология программирования»
Форма подготовки (очная)

Владивосток
2015

Паспорт ФОС

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК7 Способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений	Знает	основные положения и концепции объектно-ориентированного программирования
	Умеет	использовать методы объектно-ориентированного программирования при создании программных систем
	Владеет	методами, способами и программными средствами для разработки объектно-ориентированных программ
ОПК8 Способность использовать знания методов проектирования и производства программного продукта, принципов построения, структуры и приемов работы с инструментальными средствами, поддерживающими создание программного обеспечения	Знает	Методы проектирования объектно-ориентированных приложений
	Умеет	Проектировать требуемый набор классов и методов работы с ними при создании объектно-ориентированных приложений
	Владеет	Навыками разработки объектно-ориентированных программных средств по проекту
ОПК11 Готовность использовать навыки выбора, проектирования, реализации, оценки качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях	Знает	Методы оценки качества проекта объектно-ориентированных приложений
	Умеет	Проектировать требуемый набор тестов для оценки качества объектно-ориентированных приложений
	Владеет	Навыками создания пакета тестов для оценки качества объектно-ориентированных приложений
ПК2 готовность к использованию основных моделей информационных технологий и способов их применения для решения задач в предметных областях	Знает	принципы наследования, инкапсуляции и полиморфизма, положенные в разработку объектно-ориентированных языков
	Умеет	использовать принципы наследования, инкапсуляции и полиморфизма при создании объектно-ориентированных приложений
	Владеет	методами проектирования и разработки программ, используя полиморфные функции и операции, разные типы наследования и создавая методы доступа к объектам разных классов (инкапсуляция)

№ п/п	Контролируемые разделы/темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства - наименование	
				текущий контроль	промежуточная аттестация
1	Объекты и классы С++.	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 1	Зачет
2	Работа с классами в С++.	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 2	Зачет
3	Классы и динамическое выделение памяти	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 3	Зачет
4	Наследование	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 4	Зачет
5	Дружественные классы	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 5	Зачет
6	Потоковый ввод и вывод. Iostream.	ОПК7 ОПК8 ОПК11 ПК2	Знает Умеет Владеет	УО1 собеседование ПР2 контрольная работа ПР-6 лабораторная работа 6	Зачет

Шкала оценивания уровня сформированности компетенций

Код и формулировка компетенции	Этапы формирования компетенции		критерии	показатели
ОПК7 Способность использовать знания основных концептуальных положений	знает (пороговый уровень)	основные положения и концепции объектно-ориентированного программирования	Знание понятий и концепций наследования, инкапсуляции, полиморфизма	Способность дать ответы на вопросы
	умеет (продвинутый)	использовать методы объектно-ориентированного	Умение проектировать и программироват	Наличие созданных при выполнении лабораторных

функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений		программирования при создании программных систем	ь с использованием объектно-ориентированного языка	работ программ
	владеет (высокий)	методами, способами и программными средствами для разработки объектно-ориентированных программ	Владение методами разработки программной системы с использованием языкового процессора объектно-ориентированного языка	Наличие компьютерных программ для языкового процессора/, созданных при выполнении лабораторных работ
ОПК8 Способность использовать знания методов проектирования и производства программного продукта, принципов построения, структуры и приемов работы с инструментальными средствами, поддерживающими создание программного обеспечения	знает (пороговый уровень)	Методы проектирования объектно-ориентированных приложений	Знание методов проектирования структуры классов и операций работы с объектами классов	Способность дать ответы на вопросы
	умеет (продвинутый)	Проектировать требуемый набор классов и методов работы с ними при создании объектно-ориентированных приложений	Умение использовать методы проектирования объектно-ориентированных программ	Наличие созданных проектов
	владеет (высокий)	Навыками разработки объектно-ориентированных программных средств по проекту	Владение методами разработки объектно-ориентированных программ в некоторой среде программирования	Наличие созданных программ
ОПК11 Готовность использовать навыки выбора, проектирова	знает (пороговый уровень)	Методы оценки качества проекта объектно-ориентированных приложений	Знание методов проектирования тестовых ситуаций и тестов	Способность дать ответы на вопросы

ния, реализации, оценки качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях	умеет (продвинутый)	Проектировать требуемый набор тестов для оценки качества объектно-ориентированных приложений	Умение определить тестовые ситуации и необходимый набор тестов для проверки качества создаваемых программ	Наличие проектов тестов
	владеет (высокий)	Навыками создания пакета тестов для оценки качества объектно-ориентированных приложений	Умение записать тестовые данные средствами объектно-ориентированного языка	Наличие созданных тестов
ПК2 готовность к использованию основных моделей информационных технологий и способов их применения для решения задач в предметных областях	знает (пороговый уровень)	принципы наследования, инкапсуляции и полиморфизма, положенные в разработку объектно-ориентированных языков	Знание понятий и концепций наследования, инкапсуляции, полиморфизма	Способность дать ответы на вопросы
	умеет (продвинутый)	использовать принципы наследования, инкапсуляции и полиморфизма при создании объектно-ориентированных приложений	Умение проектировать программы, используя одиночное или множественное наследование, полиморфизм, умение проектировать методы доступа к элементам структуры класса (инкапсуляция)	Наличие проектов
	владеет (высокий)	методами проектирования и разработки программ, используя полиморфные функции и операции, разные типы наследования и создавая методы доступа к объектам разных классов (инкапсуляция)	Владение методами создания объектно-ориентированных программ, использующих разные механизмы	Наличие созданных программ

Методические рекомендации, определяющие процедуры оценивания результатов освоения дисциплины

Промежуточная аттестация студентов. Промежуточная аттестация студентов проводится в соответствии с локальными нормативными актами ДВФУ, и является обязательной.

По дисциплине предусмотрен зачет, который проводится в устной либо письменной форме.

Оценочные средства для промежуточной аттестации

Вопросы к зачету

1. **Вопросы по общей теории объектного подхода:**
    1. Эволюция методологий программирования. Парадигмы программирования.
    2. Основные принципы объектного подхода. Абстрагирование.
    3. Основные принципы объектного подхода. Инкапсуляция.
    4. Основные принципы объектного подхода. Модульность.
    5. Основные принципы объектного подхода. Иерархия.
    6. Основные принципы объектного подхода. Типизация.
    7. Объект с точки зрения ООП. Состояние. Поведение.
    8. Объект с точки зрения ООП. Идентичность и жизненный цикл объектов.
    9. Объект с точки зрения ООП. Взаимоотношения между объектами.
    10. Классы. Природа классов. Метамодель. Инстанцирование.
    11. Классы. Структура класса. Абстрактные классы и интерфейсы.
    12. Классы. Принцип подстановки Лисковой. Принцип разделения интерфейсов.
    13. Классы. Средства UML для построения диаграмм классов.
    14. Классы. Отношения между классами. Ассоциация и агрегация.
    15. Классы. Иерархии классов. Зависимость.
2. **Вопросы по C++:**
    1. Модель памяти и структура программы. Классы памяти. Ссылки.
    2. Средства абстракции C++. Структура класса. Статические члены и их инициализация
    3. Средства инкапсуляции C++. Инкапсуляция и наследование. Друзья.

4. Модульность, раздельная компиляция, пространства имен, using директива.
5. Представление иерархических отношений. Наследование.
6. Представление иерархических отношений. Агрегация. Зависимость по времени жизни.
7. Правила преобразования типов в C++. Параметрический и виртуальный полиморфизм.
8. C++: средства реализации состояния объектов; реализация поведения.
9. Перегрузка операторов.
10. Жизненный цикл объекта. Инициализация массивов. Конструкторы и деструкторы. Порядок вызова конструкторов и деструкторов при наследовании.
11. Варианты реализации отношения клиент-сервер. Объекты при передаче параметров и возврате из методов.
12. Исключения в C++. Обработка исключений. Умные указатели.
13. Шаблоны классов и шаблоны функций. Специализация.
14. Основы STL. Структура и назначение. Контейнеры.
15. Основы STL. Аллокаторы и итераторы.

Текущая аттестация студентов. Текущая аттестация проводится в форме собеседования (устного опроса) для проверки теоретических знаний, а также в форме защиты проекта, выполняемого в рамках самостоятельной работы параллельно с лабораторными работами и осуществляется ведущим преподавателем.

Объектами оценивания выступают:

– степень усвоения теоретических знаний - оценивается в форме собеседования и контрольных работ;

– уровень овладения практическими умениями и навыками – оценивается в форме защиты индивидуального заданий (проектов), выполняемых в рамках лабораторных работ.

Студенты получают индивидуальные задания. В процессе их выполнения должны быть разработаны: формальная постановка задачи, алгоритмы ее решения, написана программа на языке программирования C++, для которой созданы тесты. Правильность работы программы демонстрируется с помощью созданных тестов. Преподаватель вправе задать свои значения входных данных.

Оценочные средства для текущей аттестации

Критерии оценки устного ответа

– **100-85 баллов** - если ответ показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа; умение приводить примеры современных проблем изучаемой области.

– **85-76 баллов** - ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна - две неточности в ответе.

– **75-61 балл** - оценивается ответ, свидетельствующий в основном о знании процессов изучаемой предметной области, отличающийся недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа; неумение привести пример развития ситуации, провести связь с другими аспектами изучаемой области.

– **60-50** баллов - ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа; незнание современной проблематики изучаемой области.

Вопросы к устному ответу (собеседованию)

1. Модель памяти и структура программы. Классы памяти. Ссылки.
2. Средства абстракции C++. Структура класса. Статические члены и их инициализация
3. Средства инкапсуляции C++. Инкапсуляция и наследование. Друзья.

4. Модульность, раздельная компиляция, пространства имен, using директива.
5. Представление иерархических отношений. Наследование.
6. Представление иерархических отношений. Агрегация. Зависимость по времени жизни.
7. Правила преобразования типов в C++. Параметрический и виртуальный полиморфизм.
8. C++: средства реализации состояния объектов; реализация поведения.
9. Перегрузка операторов.
10. Жизненный цикл объекта. Инициализация массивов. Конструкторы и деструкторы. Порядок вызова конструкторов и деструкторов при наследовании.
11. Варианты реализации отношения клиент-сервер. Объекты при передаче параметров и возврате из методов.
12. Исключения в C++. Обработка исключений. Умные указатели.
13. Шаблоны классов и шаблоны функций. Специализация.
14. Основы STL. Структура и назначение. Контейнеры.
15. Основы STL. Аллокаторы и итераторы.

Контрольные работы

1. Разработать класс Дробное число со знаком (Fractions). Число должно быть представлено двумя полями: целая часть - длинное целое со знаком, дробная часть - беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения. В функции main проверить эти методы.
2. Разработать класс для работы с текстовым файлом как со строкой. Реализовать перегрузку следующих операторов: =, [], ==, . Реализовать методы: substr, strpos.
3. Описать базовый класс Строка. Обязательные поля класса:
   указатель pChar хранит адрес динамически выделенной памяти для размещения символов строки;
   значение типа word хранит длину строки в байтах.
   Реализовать обязательные методы следующего назначения:
   a. конструктор без параметров;
   b. конструктор, принимающий в качестве параметра строковый литерал; п конструктор, принимающий в качестве параметра символ;
   c. метод получения длины строки;

d. метод очистки строки (делает строку пустой);
e. деструктор.

Описать производный от Строка класс Комплексное число.
Строки данного класса должны состоять |из двух полей разделочных символом i. Первое поле задает значение действительной части числа, а второе - значение мнимой. Каждое из полей может содержать только символы десятичных цифр и символы - и +, определяющие знак числа. Символы - или + могут находиться только в первой позиции числа, причем символ + может отсутствовать, в этом случае число считается положительным. Если в составе инициализирующей строки будет встречен любой символ, отличный от допустимых, объект класса Комплексное число должен принимать нулевое значение. Примеры строк: 33il2, -7U00, +5i-21.
Для класса Комплексное_число определить следующие методы:

a. проверка на равенство;
b. сложение чисел;
c. умножение чисел.

Написать тестовую программу, которая демонстрирует работу методов базового и производного классов.

4. Описать шаблонный класс List для работы с однонаправленными списками в динамической памяти. Для объектов класса List определить операции проверки списка на пустоту, добавления элемента в начато списка, в конец списка, подсчет числа вхождений элемента в список, удаление элемента из списка. Продемонстрировать работу с шаблонным классом для списка с целыми элементами и с элементами-строками.

Критерии оценки программы по лабораторным работам (проектов)

– 100-86 баллов выставляется, если содержание и составляющие части соответствуют выданному заданию. Продемонстрировано владение навыками разработки, тестирования программ на языке программирования C++. Программа правильно работает на всех наборах входных данных. Текст программы содержит комментарии.

– 85-76 - баллов выставляется, если при выполнении задания допущено не более одной ошибки. Продемонстрировано владение навыками разработки программ на языке программирования C++. Программа правильно работает не на всех наборах входных данных (90%). Текст программы содержит комментарии.

– 75-61 балл выставляется, если при выполнении задания допущено не более двух ошибок. Продемонстрировано знание методов разработки программ на языке программирования С++. Программа правильно работает не на всех наборах входных данных (70%). В тексте программы комментарии отсутствуют.

60-50 баллов - если структура и содержание задания не соответствуют требуемым

Шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Примерные задания к лабораторным работам

ЛАБОРАТОРНАЯ РАБОТА 1. ОБЪЕКТЫ И КЛАССЫ С++

Получение базовых знаний в области объектно-ориентированного программирования, изучение основных понятий и парадигм: определение класса для заданной задачи, программирование методов класса, типы конструкторов, прототипы функций.

Задание 1. Определите класс для представления банковского счета. Данные-члены должны включать имя вкладчика, номер счета (используйте строку) и баланс. Функции-члены должны позволять следующее:

- создание объекта и его инициализация;
- отображение имени вкладчика, номера счета и баланса;
- добавление на счет суммы денег, переданной в аргументе;
- снятие суммы денег, переданной в аргументе.

Напишите программу для иллюстрации всех его возможностей.

Задание 2. Пусть имеется определение следующего простого класса:

```
class Person {
private:
static const LIMIT = 25; string lname;        // фамилия
char fname [LIMIT] ;     // имя
public:
Person () { lname = fname[0] = * \0 *; }      // #1
```

```
Person (const string & In, const char * fn = "Heyyou"); // #2
// Следующие методы отображают lname и fname void Show() const;
// формат: имя фамилия
void FormalShow() const; // формат: фамилия, имя
};
```

(В нем используется объект string и символьный массив, так что вы сможете сравнить применение этих двух форм.) Напишите программу, которая дополнит реализацию за счет предоставления кода для пока еще не определенных методов. В программе, использующей класс, должны также присутствовать вызовы трех возможных конструкторов (без аргументов, с одним аргументом, с двумя аргументами) и двух методов отображения. Ниже приведен пример применения этих конструкторов и методов:

```
Person one; // используется конструктор по умолчанию
Person two("Smythecraft");      // используется конструктор #2
//с одним аргументом по умолчанию Person three("Dimwiddy", "Sam");
// используется конструктор #2,
// без аргументов по умолчанию
one.Show(); cout « endl; one.FormalShow();
// и т.д. для объектов two и three.
```

Задание 3. Пусть имеется следующее объявление структуры:

```
struct customer {
char fullname[35]; double payment;
};
```

Напишите программу, которая будет добавлять структуры заказчиков в стек и удалять их из стека, представленного объявлением класса Stack. Всякий раз, когда заказчик удаляется из стека, его зарплата должна добавляться к промежуточной сумме и по этой сумме выдаваться отчет. На заметку: вы должны иметь возможность пользоваться классом Stack без изменений; просто поменяйте объявление typedef, чтобы Item был типом customer вместо unsigned long.

Задание 4. Пусть имеется следующее объявление класса:

```
class Move {
private:
double x;
```

```
double y;
public:
Move (double a = 0, double b = 0) ; // устанавливает x, y в a, b
showmove() const;        // отображает текущие значения x, y
Move add (const Move & m) const;
// Эта функция добавляет x из m к x вызывающего объекта,
// чтобы получить новое значение x,
// Добавляет y из ш к y вызывающего объекта, чтобы получить новое //
значение y, присваивает инициализированному объекту значения x, y
//и возвращает его
reset (double a = 0, double b = 0) ; // сбрасывает x, y в a, b
};
```

Создайте определения функций-членов и напишите программу, которая использует этот класс.

Задание 5. Плорг из Бетельгейзе обладает следующими свойствами:

Данные

плорг имеет имя не длиннее 19 символов;

плорг имеет индекс удовлетворенности (contentment index — CI), выражаемый целым числом.

Операции

новый плорг начинает существование с именем и индексом CI равным 50;

индекс CI плорга может изменяться;

плорг может сообщать свое имя и индекс CI;

по умолчанию плорг имеет имя "Plorga".

Напишите объявление класса Plorg (включая данные-члены и прототипы функций-членов), который представляет плорга. Напишите определения функций-членов. Напишите короткую программу, демонстрирующую все средства класса Plorg.

Задание 6. Простой список можно описать следующим образом:

простой список может содержать ноль или более элементов определенного типа;

можно создавать пустой список;

можно добавлять элемент в список;

можно определять, пуст ли список;

можно определять, полон ли список.

можно посетить каждый элемент списка и выполнить над ним определенное действие.

Спроектируйте класс List для представления этого абстрактного типа. Вы должны подготовить заголовочный файл list.h с объявлением класса и файл list.cpp с реализацией его методов. Вы должны также написать короткую программу, которая будет использовать полученный класс.

ЛАБОРАТОРНАЯ РАБОТА 2. РАБОТА С КЛАССАМИ В C++

Получение практических навыков в области перегрузки операторов.

Разработка классов для работы с данными разных типов: стек, список, вектор, комплексные числа и т.д. Перегрузка операций. Дружественные функции.

Задание 1. Приведенная ниже программа моделирует известную задачу случайного блуждания. Идея заключается в том, что вы помещаете кого-то в исходную точку. Он начинает двигаться, но направление с каждым шагом случайным образом изменяется по отношению к направлению на предыдущем шаге. Вот как выглядит одна из формулировок этой задачи: сколько шагов нужно сделать этому персонажу, чтобы удалиться, скажем, на 50 футов от исходной точки? В терминах векторов это означает сложение множества случайно ориентированных векторов до тех пор, пока сумма не превысит 50 футов.

```
// randwalk.cpp — использование класса Vector // Компилировать вместе с файлом vect.cpp #include <iostream>
#include <cstdlib>// прототипы rand(), srand()
#include <ctime>  // прототип time()
#include "vect.h" int main()

{
  using namespace std; using VECTOR::Vector;
  srand(time(0)); // начальное значение для генератора случайных чисел
  double direction;
  Vector step;
  Vector result(0.0, 0.0); unsigned long steps = 0; double target; double dstep;
  cout « "Enter target distance (q to quit) : ";
```

```
// Ввод заданного расстояния (q для завершения) while (cin » target)
{
 cout « "Enter step length: "; // ввод длины шага if (! (cin » dstep) ) break;
 while (result.magval () < target)
 {
  direction = rand() % 360;
  step.reset(dstep, direction, Vector::POL);
  result = result + step; steps++;
 }
 cout « "After " « steps « " steps, the subject "
"has the following location:\n"; cout « result « endl; // вывод позиции после
steps шагов result.polar_mode(); cout « " or\n" « result « endl; cout « "Average
outward distance per step = "
 « result.magval()/steps « endl; // вывод среднего расстояния на один
шаг steps = 0;
 result.reset (0.0, 0.0);
 cout « "Enter target distance (q to quit): ";
 // Ввод заданного расстояния (q для завершения)
}
cout « "Bye!\n"; cin.clear();
while (cin.getO != ' \n ') continue; return 0;
```

Модифицируйте программу так, чтобы обеспечить запись в файл последовательных позиций при случайном блуждании. Каждая позиция должна помечаться номером шага. Также программа должна записывать в файл начальные условия (целевое расстояние и длину шага) и суммарные результаты.

Задание 2. Ниже приведен заголовок класса Vector:

```
// vect.h — класс Vector с операцией « и поддержкой режима координат
#ifndef VECT0R_H_
#define VECT0R_H_
#include <iostream> namespace VECTOR {
 class Vector
 {
 public:

  enum Mode {RECT, POL};
  // RECT — для режима прямоугольных координат, POL — для
режима полярных координат private:
```

```
    // горизонтальное значение // вертикальное значение // длина вектора
    // направление вектора в градусах // RECT или POL
    // Закрытые методы для установки значений void set_mag(); void set_ang(); void set_x(); void set_y(); public:

    Vector();

    Vector (double nl, double n2, Mode form = RECT); void reset (double nl, double n2, Mode form = RECT); -Vector();
    // сообщает значение x // сообщает значение y
    сообщает модуль сообщает угол
    POL RECT

    // Перегрузка операций
    Vector operator+(const Vector & b) const;
    Vector operator-(const Vector & b) const;
    Vector operator-() const;
    Vector operator*(double n) const;
    // Друзья

    friend Vector operator*(double n, const Vector & a) ; friend std::ostream &

    operator« (std: :ostream & os, const Vector & v) ;
  };

} // конец пространства имен VECTOR #endif
```

Модифицируйте заголовок класса Vector так, чтобы модуль и направление вектора больше не хранились в виде компонентов данных. Вместо этого они должны вычисляться по требованию при вызове методов magval () и angval (). Вы должны оставить открытый интерфейс без изменений (те же открытые методы с теми же аргументами), но изменить закрытую часть, включая некоторые из закрытых методов и их реализации. Протестируйте модифицированную версию с помощью программы из ЛР 1, которая должна остаться неизменной, поскольку открытый интерфейс класса Vector не менялся.

Задание 3. Ниже представлено описание и реализация класса Time:

```
// time.h -- класс Time с друзьями

#ifndef MYTIME3_H_
#define MYTIME3_H_
#include <iostream> class Time {
private:
  int hours; int minutes; public:
  Time();
  Time (int h, int m = 0) ;
  void AddMin (int m) ;
  void AddHr(int h) ;
  void Reset (int h = 0, int m = 0) ;
  Time operator+(const Time & t) const;
  Time operator-(const Time & t) const;
  Time operator*(double n) const;
  friend Time operator*(double m, const Time & t)
  { return t * m; }   // встроенное определение
  friend std::ostream & operator«(std::ostream & os, const Time & t) ;
};
#endif

// time.cpp — реализация методов класса Time #include "mytime3.h"

Time::Time()
{
  hours = minutes = 0;
}
Time::Time(int h, int m )
{
  hours = h; minutes = m;
}
void Time : : AddMin (int m)
{
  minutes += m;
  hours += minutes / 60;
  minutes %= 60;
}
void Time::AddHr(int h)
```

```
{
 hours += h;
}
void Time::Reset (int h, int m)
{
 hours = h; minutes = m;
}
Time Time::operator+(const Time & t) const
{
 Time sum;
 sum.minutes = minutes + t. minutes;
 sum.hours = hours + t.hours + sum.minutes / 60;
 sum.minutes %= 60;
 return sum;
}
Time Time:: operator-(const Time & t) const
{
 Time diff; int totl, tot2;
 totl = t.minutes + 60 * t.hours; tot2 = minutes + 60 * hours; diff .minutes =
(tot2 - totl) % 60; diff .hours = (tot2 — totl) / 60; return diff;
}
Time Time::operator*(double mult) const
{
 Time result;
 long totalminutes = hours * mult * 60 + minutes * mult; result.hours =
totalminutes / 60; result.minutes = totalminutes % 60; return result;
}
std: rostream & operator« (std: :ostream & os, const Time & t)
{
 os « t.hours « " hours, " « t.minutes « " minutes"; return os;
}
```

Перепишите класс Time так, чтобы все перегруженные операции были реализованы с использованием дружественных функций.

Задание 4. Ниже представлено описание и реализация класса Stonewt:

```
// stonewt.h — определение класса Stonewt

#ifndef STONEWT_H_
```

```
#define STONEWT_H_

class Stonewt

private:
enum {Lbs_per_stn =14};        // фунтов на стоун
int stone;     // полных стоунов
double pdsleft;      // дробное число фунтов
double pounds;      // общий вес в фунтах
public:
Stonewt(double lbs);       // конструктор для значения в фунтах
Stonewt(int stn, double lbs);     // конструктор для значения в стоунах и фунтах
Stonewt ();  // конструктор по умолчанию
-Stonewt ();
void show_lbs() const;    // отображение веса в формате фунтов
void show_stn() const;    // отображение веса в формате стоунов
};

#endif

// stonewt.cpp — методы класса Stonewt
#include <iostream> using std::cout;
#include "stonewt.h"

// Конструирует объект Stonewt из значения типа double
Stonewt::Stonewt(double lbs)
{
stone = int (lbs) / Lbs_per_stn; // целочисленное деление
pds_left = int (lbs) % Lbs_per_stn + lbs - int (lbs); pounds = lbs;
}

// Конструирует объект Stonewt из стоунов и значения типа double
Stonewt::Stonewt (int stn, double lbs)
{
stone = stn; ‘ pds_left = lbs;
pounds = stn * Lbs_per_stn +lbs;
}
```

```
Stonewt::Stonewt ()        // конструктор по умолчанию, wt = 0
{
stone = pounds = pds_left = 0;
}

Stonewt::~Stonewt()        // деструктор
{
}

// Отображение веса в стоунах
void Stonewt::show_stn () const {
cout « stone << " stone, " « pds_left << " pounds\n";
}

// Отображение веса в фунтах
void Stonewt::show_lbs() const {
cout « pounds « " pounds\n";
}
```

Перепишите класс Stonewt так, чтобы перегружались все шесть операций сравнения. Операции должны сравнивать члены pounds и возвращать значение типа bool. Напишите программу, которая объявляет массив из шести объектов Stonewt с инициализацией в объявлении первых трех из них. Затем программа должна в цикле читать значения, используемые для установки остальных трех элементов массива. После этого программа должна вывести самый маленький элемент, самый большой, а также количество элементов, которые больше или равны 11 стоунам. (Простейший подход предполагает создание объекта Stonewt, инициализированного 11 стоунами, и сравнение с ним других объектов.)

Задание 5. Комплексное число состоит из двух частей — вещественной и мнимой. Один из способов записи такого числа выглядит как (3.0, 4.0). Здесь 3.0 — вещественная часть, а 4.0 — мнимая. Предположим, что a = (A, Bi) и c = (C, Di). Ниже представлены некоторые операции с комплексными числами:

сложение: a + c = (A + C, (B + D)i)
вычитание: a - c = (A - C, (B - D)i)
умножение: a*c = (AxC - BxD, (AxD + BxC)i)
умножение (x — вещественное число): xxc = (xxC, xxDi)

сопряжение: ~a = (A, - Bi)
Определите класс complex так, чтобы следующая программа могла использовать его с корректными результатами:

```
#include <iostream> using namespace std;
#include "complexO.h" // во избежание конфликта с complex.h int main()
{
complex a (3.0, 4.0);        // инициализация значением (3,4i)
complex c;
cout << "Enter a complex number (q to quit) :\n";
// Ввод комплексного числа (q для завершения) while (cin » c)
{
cout << "c is " << c << '\n';        // значение c
cout << "complex conjugate is " << ~c << '\n';
// значение сопряженного числа cout << "a is " << a << '\n' ; 9 // значение
a
cout << "a + c is " « a + c « '\n'; // значение a + c
cout << "a — c is " << a — c << '\n'; // значение a — c
cout << "a * c is " << a * c << '\n'; // значение a * c
cout << "2 •* c is " « 2 * c << ' \n '; // значение 2 * c
cout << "Enter a complex number (q to quit) :\n";
}
cout « "Done!\n"; return 0;
}
```

Не забывайте, что вы должны перегрузить операции << и >>. В стандарте C++ уже присутствует поддержка комплексных чисел — и намного более развитая, чем в этом примере — в заголовочном файле complex, поэтому во избежание конфликтов назовите свой файл complex.h. Используйте const там, где это оправдано.
Ниже показан пример выполнения этой программы:
Enter a complex number (q to quit) : real: 10 imaginary: 12 c is (10,12i)
complex conjugate is (10,-121) a is (3,4i) a + c is (13,16i) a - c is (-7,-8i) a * c is (-18, 76i)
2 * c is (20, 24i)
Enter a complex number (q to quit) : real: q Done!
Обратите внимание, что благодаря перегрузке, cin >> c теперь запрашивает ввод вещественной и мнимой частей комплексного числа.

ЛАБОРАТОРНАЯ РАБОТА 3. КЛАССЫ И ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

Получение практических навыков в выделении динамической памяти под объекты классов.

Задание 1. Имеется следующее объявление класса:

```
class Cow {
char name[20]; char * hobby; double weight; public:
Cow();
Cow (const char * nm, const char * ho, double wt) ;
Cow(const Cow c&) ;
-Cow();
Cow & operator= (const Cow & c) ;
void ShowCow() const; // отображение всех данных cow
};
```

Напишите реализацию для этого класса и короткую программу, использующую все функции-члены.

Задание 2. Усовершенствуйте объявление класса String (т.е. замените stringl.h на string2 .h), выполнив перечисленные ниже действия.

а. Перегрузите операцию + для объединения двух строк в одну.

б. Напишите функцию-член stringlow (), которая преобразует все буквенные символы в строке в нижний регистр. (Не забудьте о семействе cctype символьных функций.)

в. Напишите функцию-член stringupO , которая преобразует все буквенные символы в строке в верхний регистр.

г. Напишите функцию-член, которая принимает аргумент типа char и возвращает количество раз, которое символ появляется в строке.

Проверьте работу полученного класса в следующей программе:

```
// pe12_2.cpp
#include <iostream>
using namespace std;
#include "string2.h"

int main ()
{
 String si (" and I am a C++ student.");
  String s2 = "Please enter your name: "; // ввод имени
  String s3;
```

```
cout << s2;// перегруженная операция <<
cin >> s3;  // перегруженная операция >>

s2 = "My name is " + s3;        // перегруженные операции =, +

cout « s2 << ". \n";
s2 '= s2 + si;

s2.stringup ();      // преобразование строки в верхний регистр

cout « "The string\n" « s2 « "\ncontains " « s2.has('A')

« " 'A' characters in it.\n"; si = "red";          // String(const char *) ,

// тогда

String & operator=(const Strings) String rgb[3] = { String (si), String ("green"), String("blue")}; cout « "Enter the name of a primary color for mixing light: "; // ввод цвета
String ans; bool success = false; while (cin >> ans)
ans.stringlow();   // преобразование строки в нижний регистр

for (int i = 0; i < 3; i ++)
{
  if (ans == rgb[i])         // перегруженная операция ==
  {
    cout « "That's right!\n";
    success = true;
    break;
    1
  }

  if (success) break; else
  cout « "Try again! \n";
}

cout << "Bye\n"; return 0;
```

Вывод программы должен выглядеть приблизительно так:
Please enter your name: Fretta Farbo My name is Fretta Farbo.
The string
MY NAME IS FRETTA FARBO AND I AM A C++ STUDENT.
contains 6 'A' characters in it.
Enter the name of a primary color for mixing light: yellow Try again!
BLUE
That's right!
Bye

Задание 3. Перепишите класс Stock, описанный ниже, чтобы он использовал для хранения названий пакетов акций непосредственно динамически выделенную память, а не объекты класса string. Кроме того, замените функцию-член show() перегруженным определением operator<< ().

```
// stock.h — дополненная версия

#ifndef STOCK_H_
#define STOCK_H_
#include <string>
class Stock {
private:
  std::string company; int shares; double share_val;
  double total_val;
  void set_tot() { total_val = shares * share_val; } public:
  Stock();     // конструктор по умолчанию
  Stock(const std::string & co, long n = 0, double pr = 0.0);
  ~Stock();   // деструктор
  void buy(long num, double price);
  void sell(long num, double price);
  void update(double price);
  void show()const;
  const Stock & topval(const Stock & s) const;
};
#endif

// stock.cpp -- дополненная версия
#include <iostream>
#include "stock20.h"
```

```
// Конструкторы
Stock::Stock()       // конструктор по умолчанию
{
  company = "no name"; shares = 0; share_val = 0.0; total_val = 0.0;
}
Stock::;Stock(const std::string & co, long n, double pr)
{
  company = co; if (n < 0)
  {
    std::cout << "Number of shares can't be negative; "
    « company « " shares set to 0.\n"; shares = 0;
  }
  else
  shares = n; share_val = pr; set_tot();
}
// Деструктор
Stock::~Stock()     // деструктор, не выводящий сообщений
{
}
// Другие методы
void Stock::buy(long num, double price)
{
  if (num < 0)
  {
    std::cout << "Number of shares purchased can't be negative. "
    << "Transaction is aborted.\n";
    else
    {
      shares += num; share_val = price; set_tot();
    }
  }
  void Stock::sell(long num, double price)
  {
    using std::cout; if (num < 0)
    {
      cout « "Number of shares sold can't be negative. " « "Transaction is
aborted.\n";
    }
    else if (num > shares)
```

```
    {'
cout « "You can't sell more than you have! "
« "Transaction is aborted.\n";
}
else
{
shares -= num; share_val = price; set_tot();
}
void Stock::update(double price)
{
share_val = price; set_tot();
}
void Stock::show () const
{
using std::cout; using std::ios_base;

// Установка формата в #.###

ios_base::fmtflags orig =
cout.setf(ios_base::fixed, ios_base::floatfield); std::streamsize prec =
cout.precision(3); cout « "Company: " « company
« " Shares: " << shares « '\n'; cout « " Share Price: $" << share_val;
// Установка формата в #.## cout.precision (2);
cout « " Total Worth: $" « total_val « ' \n' ;
// Восстановление исходного формата cout.setf(orig, ios_base::floatfield);
cout.precision(prec);
const Stock & Stock: : topval (const Stock & s) const
{
if (s.total_val > total_val) return s; else
return *this;
}
```

Задание 4. Имеется следующий вариант класса Stack:

```
// stack.h — объявление класса для АТД стека
typedef unsigned long Item;
class Stack
private:
enum {MAX =10};        // константа, специфичная для класса
```

```
Item * pitems;      // хранит элементы стека
int size;       // количество элементов в стеке
int top;        // индекс для верхнего элемента стека
public:
Stack (int n = 10); // создает стек с п элементами
Stack(const Stack & st) ;
-Stack ();
bool isemptyO const; bool isfull() const;
// push() возвращает значение false, если стек уже полный,
//и true в противном случае
bool push(const Item & item); // добавление элемента в стек // pop () возвращает значение false, если стек уже пустой,
// и true в противном случае
bool pop(Item & item);   // извлечение элемента из стека
Stack & operator= (const Stack & st) ;
```

Как понятно из закрытых членов, данный класс использует динамически выделенный массив для хранения элементов стека. Перепишите методы для соответствия новому представлению и напишите программу, которая демонстрирует работу всех методов, включая конструктор копирования и операцию присваивания.

Задание 5. Исследование банка “Bank of Heather” показало, что клиенты банкомата це ожидают в очереди более одной минуты. С помощью модели приведенной ниже найдите количество клиентов за час, которое приводит к среднему времени ожидания, равному одной минуте. (Применяйте по меньшей мере 100-часовый период моделирования.)

```
// queue.h — интерфейс для очереди

#ifndef QUEUE_H_
#define QUEUE_H_

// Очередь, содержащая элементы Customer class Customer {
private:
long arrive;  // момент появления клиента
int processtime;     // время обслуживания клиента

public:
Customer() { arrive = processtime =0; }
```

```
void set(long when);
long when() const { return arrive; }
int ptime() const { return processtime; }
};
typedef Customer Item;
class Queue
{

private:
// Определения области действия класса
// Node - вложенная структура, локальная для данного класса
struct Node { Item item; struct Node * next;}; enum {Q_SIZE = 10};
// Закрытые члены класса

Node * front;        // указатель на начало Queue
Node * rear;// указатель на конец Queue
int items;     // текущее количество элементов в Queue
const int qsize;      // максимальное количество элементов в Queue
// Упреждающие объявления для предотвращения открытого копирования

Queue(const Queue & q) : qsize (0) { }
Queue & operator=(const Queue & q) { return *this; } public:
Queue(int qs = Q_SIZE);        // создание очереди с предельным размером qs

~Queue ();

bool isemptyO const; bool isfullO const; int queuecount () const;

bool enqueue(const Item &item); // добавление элемента в конец
bool dequeue(Item &item);      // удаление элемента из начала

};

#endif
```

Задание 6. Банк “Bank of Heather” интересуется, что произойдет, если установить второй банкомат. Модифицируйте код ЛР 5 так, чтобы

поддерживались две очереди. Сделайте так, чтобы клиент присоединялся к первой очереди, если в ней меньше людей, и ко второй — в противном случае. Найдите количество клиентов за час, которое приводит к среднему времени ожидания, равному одной минуте. (Обратите внимание, что это — нелинейная задача, т.е. удвоение количества банкоматов не удваивает количество клиентов, которые могут быть обслужены за час с максимальным ожиданием в одну минуту.)

ЛАБОРАТОРНАЯ РАБОТА 4. НАСЛЕДОВАНИЕ

Получение практических навыков в области одиночного и множественного наследования классов. Базовый класс. Виртуальные методы.

Задание 1. Начните со следующего объявления класса:

```
// Базовый класс
class Cd {    // представляет компакт-диск
private:
char performers[50]; char label[20];
int selections;        // количество сборников
double playtime;   // время воспроизведения в минутах
public:
Cd(char * si, char * s2, int n, double x) ;
Cd (const Cd & d) ;
Cd () ;
~Cd() ;
void Report() const;        // выводит все данные о компакт-диске
Cd & operator= (const Cd & d) ;
};
```

Породите класс Classic, добавив массив членов char, которые будут хранить строку с названием основного произведения на компакт-диске. Если необходимо, чтобы какие-то функции в базовом классе были виртуальными, измените объявление базового класса. Если объявленный метод не нужен, удалите его из определения. Протестируйте результат с помощью следующей программы:

```
#include <iostream>
using namespace std;
#include "classic.h"        // будет содержать #include cd.h
void Bravo(const Cd & disk); int main()
{
Cd cl("Beatles", "Capitol", 14, 35.5);
```

```
Classic c2 = Classic("Piano Sonata in B flat, Fantasia in C",
"Alfred Brendel", "Philips", 2, 57.17);
Cd *pcd = &cl;
// Непосредственное использование объектов cout « "Using object directly:\n";
cl.ReportO; // использование метода Cd
c2.Report(); // использование метода Classic
// Использование указателя на объекты типа cd * cout « "Using type cd * pointer to objects:\n";
pcd->Report();      // использование метода Cd для объекта cd
pcd = &c2;
pcd->Report ();     // использование метода Classic для объекта classic
// Вызов функции с аргументом-ссылкой на Cd
cout « "Calling a function with a Cd reference argument:\n";
Bravo(cl) ;
Bravo(c2);
// Тестирование присваивания cout « "Testing assignment: ";
Classic copy; copy = c2; copy.Report() return 0;
}
void Bravo (const Cd & disk)
{
```

***Задание* 2.** Выполните задание 1, но для различных строк, используемых двумя классами, вместо массивов фиксированного размера применяйте динамическое выделение памяти.

Задание 3. Напишите программу, запрашивающую у пользователя ввод двух целых чисел. Затем программа должна вычислить и выдать сумму всех целых чисел, лежащих между этими двумя целыми. Предполагается, что меньшее значение вводится первым. Например, если пользователь ввел 2 и 9, программа должна сообщить, что сумма всех целых чисел от 2 до 9 равна 44.

Задание 4. Напишите программу, которая приглашает пользователя вводить числа. После каждого введенного значения программа должна выдавать накопленную сумму введенных значений. Программа должна завершаться при вводе 0.

Задание 5. Дарья инвестировала 1000 руб под простые 10%. Другими словами, ежегодно инвестиция должна приносить 10% инвестированной суммы, т.е. 100 руб каждый год:

прибыль = 0,10 х исходный баланс

В то же время Гликерья инвестировала 1000 руб под сложные 5%. Это значит, что прибыль составит 5% от текущего баланса, включая предыдущую накопленную прибыль:

прибыль = 0,05 х текущий баланс

Клео зарабатывает 5% от 1000 руб в первый год, что дает ей 1050. На следующий год она зарабатывает 5% от 1050, что составляет 52,5 руб, и т.д. Напишите программу, которая вычислит, сколько лет понадобится для того, чтобы сумма баланса Гликерьи превысила сумму баланса Дарьи, с отображением значений обоих балансов за каждый год.

Задание 6. Предположим, что вы продаете книгу по программированию на языке C++ для начинающих. Напишите программу, которая позволит ввести ежемесячные объемы продаж в течение года (в количестве книг, а не в деньгах). Программа должна использовать цикл, в котором выводится приглашение с названием месяца, применяя массив указателей на char (или массив объектов string, если вы предпочитаете его), инициализированный строками — названиями месяцев, и сохраняя введенные значения в массиве int. Затем программа должна найти сумму содержимого массива и выдать общий объем продаж за год.

Задание 7. Выполните задание 4, но используя двумерный массив для сохранения данных о месячных продажах за 3 года. Выдайте общую сумму продаж за каждый год и за все годы вместе.

Задание 8. Разработайте структуру по имени car, которая будет хранить следующую информацию об автомобиле: название производителя в виде строки в символьном массиве или в объекте string, а также год выпуска автомобиля в виде целого числа. Напишите программу, которая запросит пользователя, сколько автомобилей необходимо включить в каталог. Затем программа должна применить new для создания динамического массива структур car указанного пользователем размера. Далее она должна пригласить пользователя ввести название производителя и год выпуска для наполнения данными каждой структуры в массиве. И, наконец, она должна

отобразить содержимое каждой структуры. Пример запуска программы должен выглядеть подобно следующему:

Сколько автомобилей поместить в каталог? 2
Автомобиль #1:
Введите производителя: Toyota
Укажите год выпуска: 2007
Автомобиль #2:
Введите производителя: Lexus
Укажите год выпуска: 2010
Вот ваша коллекция:
2007 Toyota 2010 Lexus

Задание 9. Союз программистов-меценатов собирает коллекцию бутылочного портвейна. Для ее описания администратор союза разработал класс Port:

```
#include <iostream>
using namespace std;
class Port    //       портвейн
{
private:
char * brand;
char style[20];      //       например, tawny (золотистый),
// ruby (рубиновый), vintage (марочный)
int bottles; public:
Port(const char * br = "none", const char * st = "none", int b = 0) ;
Port(const Port & p) ;     // конструктор копирования
virtual ~Port() { delete [] brand; }
Port & operator= (const Port & p) ;
Port & operator+= (int b) ;       //       добавляет b к       bottles
Port & operator-= (int b) ;       //       вычитает b из       bottles,       если
то возможно
int BottleCount() const { return bottles; } virtual void Show() const;
friend ostream & operator« (ostream & os, const Port & p) ;
};
```

Метод Show () выводит информацию в следующем формате:
Brand: Gallo Kind: tawny Bottles: 20
Функция operator<< () представляет информацию в следующем формате (без символа новой строки в конце):
Gallo, tawny, 20

Завершив определения методов для класса Port, администратор написал производный класс VintagePort, прежде чем был уволен:

```
class VintagePort : public Port // style обязательно = "vintage"
{
private:
char * nickname;   // т.е. "The Noble", "Old Velvet" и т.д.
int year;       // год сбора
public:
VintagePort ();
VintagePort (const char * br, int b, const char * nn, int y) ;
VintagePort(const VintagePort & vp) ;
^VintagePort() { delete [] nickname; }
VintagePort & operator=(const VintagePort & vp) ; void Show() const;
friend ostream & operator«(ostream & os, const VintagePort & vp) ;
};
```

Вам поручено завершить разработку класса VintagePort.

а. Первое задание — нужно заново создать определения методов Port, т.к. предыдущий администратор уничтожил свой код.

б. Второе задание — объясните, почему одни методы переопределены, а другие нет.

в. Третье задание — объясните, почему функции operator= () и operator« () не определены как виртуальные.

г. Четвертое задание — обеспечьте определения для методов VintagePort.

ЛАБОРАТОРНАЯ РАБОТА 5. ДРУЖЕСТВЕННЫЕ КЛАССЫ

Получение практических навыков работы в области разработки дружественных классов

Задание 1. Измените приведенный ниже код так, чтобы два типа исключений были классами, производными от класса logic error, определенного в заголовочном файле <stdexcept>. Сделайте так, чтобы каждый метод what () сообщал имя функции и суть проблемы. Объекты исключений не должны содержать значение ошибки, они должны просто поддерживать метод what ().

```
// errog4.cpp — использование классов исключений
#include <iostream>
#include <cmath> // или math.h, пользователям UNIX может потребоваться флаг -lm #include "exc_mean.h"
// Прототипы функций
```

```
double hmean(double a, double b); double gmean(double a, double b) ; int main()
{
using std::cout; using std::cin; using std::endl; double x, y, z;
cout << "Enter two numbers:    // запрос на ввод двух чисел
while (cin >> x >> y)
{
try {   // начало блока try
z = hmean(x,y);
cout << "Harmonic mean of " « x << " and " « y
« " is " « z « endl;  // вывод среднего гармонического
cout << "Geometric mean of " « x « " and " « y
« " is " « gmean(x,y) « endl;     // вывод среднего геометрического
cout << "Enter next set of numbers <q to quit>: "; // ввод следующей пары чисел }       // конец блока try
catch (bad_hmean & bg) // начало блока catch

{
bg.mesg ();
cout « "Try again.\n";      // необходимо повторить попытку
continue;
catch (bad_gmean & hg)
{
cout « hg.mesgO;
cout « "Values used: " « hg.vl « ", "
« hg.v2 « endl;      // используемые знарчения
cout << "Sorry, you don't get to play any more.\n"; // завершение работы
break;
}       // конец блока catch
}
cout << "Bye!\n"; return 0;
double hmean(double a, double b)
{
if (a    b)
throw bad_hmean (a,b); return 2.0 * a * b / (a + b) ;
double gmean(double a, double b)
{
if (a < 0 | | b < 0)
throw bad_gmean (a,b); return std::sqrt(a * b) ;
```

Задание 2. Напишите программу, читающую в массив double до 10 значений пожертвований. Программа должна прекращать ввод при получении нечисловой величины. Она должна выдавать среднее значение полученных чисел, а также количество значений в массиве, превышающих среднее.

Задание 3. Напишите предшественник программы, управляемой меню. Она должна отображать меню из четырех пунктов, каждый из них помечен буквой. Если пользователь вводит букву, отличающуюся от четырех допустимых, программа должна повторно приглашать его ввести правильное значение до тех пор, пока он этого не сделает. Затем она должна выполнить некоторое простое действие на основе пользовательского выбора. Работа программы должна выглядеть примерно так:

```
Please enter one of the following choices:
c) carnivore p) pianist
t) tree      g) game
f
Please enter a c, p, t, or g: q
Please enter a c, p, t, or g: t
A maple is a tree.
```

Задание 4. Постройте программу, которая отслеживает пожертвования в Общество Защиты Влиятельных Лиц. Она должна запрашивать у пользователя количество меценатов, а затем приглашать вводить их имена и суммы пожертвований от каждого. Информация должна сохраняться в динамически выделяемом массиве структур. Каждая структура должна иметь два члена: символьный массив (или объект string) для хранения имени и переменную-член типа double — для хранения суммы пожертвования. После чтения всех данных программа должна отображать имена и суммы пожертвований тех, кто не пожалел 10 000 и более. Этот список должен быть озаглавлен меткой “Grand Patrons”. После этого программа должна выдать список остальных жертвователей. Он должен быть озаглавлен “Patrons”. Если в одной из двух категорий не окажется никого, программа должна напечатать “none”. Помимо отображения двух категорий, никакой другой сортировки делать не нужно.

Задание 5.Напишите программу, которая читает слова по одному за раз, пока не будет введена отдельная буква q. После этого программа должна сообщить количество слов, начинающихся с гласных, количество слов,

начинающихся с согласных, а также количество слов, не попадающих ни в одну из этих категорий.

Задание 6. Это задание подобно заданию 1, но исключения должны быть производными от базового класса (потомка logic_error), который хранит два значения аргументов. Исключения должны содержать метод, который выводит эти значения и имя функции, и единственный блок catch, который используется для обоих исключений. Каждое исключение должно приводить к прекращению цикла обработки.

ЛАБОРАТОРНАЯ РАБОТА 6. ПОТОКОВЫЙ ВВОД И ВЫВОД. IOSTREAM

Получение практических навыков работы с потоками ввода и вывода C++.

Задание 1. Напишите программу, которая подсчитывает количество символов вплоть до первого символа $ в строке, оставляя $ во входном потоке.

Задание 2. Напишите программу, которая копирует клавиатурный ввод (вплоть до эмулируемого конца файла) в файл, имя которого передано в командной строке.

Задание 3. Напишите программу, копирующую один файл в другой. Имена файлов программа должна получать из командной строки. Если не удается открыть файл, должно выдаваться соответствующее сообщение.

Задание 4. Напишите программу, которая открывает два текстовых файла для ввода и один для вывода. Программа должна соединять соответствующие строки входных файлов, используя в качестве разделителя пробел, и записывать результаты в выходной файл. Если один файл короче второго, остальные строки более длинного файла также должны копироваться в выходной файл. Например, предположим, что первый входной файл имеет следующее содержимое:

eggs kites donuts balloons hammers stones

А второй файл — такое:

zero lassitude finance drama

Результирующий файл должен выглядеть следующим образом:

eggs kites donuts zero lassitude balloons hammers finance drama stones