

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Смирнов Сергей Николаевич
Должность: врио ректора
Дата подписания: 16.10.2023 21:40:08
Уникальный программный ключ:
69e375c64f7e975d4e8830e7b4fcc2ad1bf35f08

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тверской государственный университет»

Утверждаю:
Руководитель ООП
Н.А. Семькина
« 4 » 09
МАТЕМАТИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ
УНИВЕРСИТЕТ

Рабочая программа дисциплины (с аннотацией)

Прикладное программирование

Специальность

10.05.01 Компьютерная безопасность

Специализация

«Математические методы защиты информации»

Для студентов 4 курса очной формы обучения

Уровень высшего образования

СПЕЦИАЛИТЕТ

Составитель:

ст. преподаватель Е.В. Тишина

Тверь, 2023

I. Аннотация

1. Наименование дисциплины (модуля) в соответствии с учебным планом Прикладное программирование

2. Цель и задачи дисциплины (модуля)

Целью изучения дисциплины "Прикладное программирование" является овладение знаниями и навыками проектирования прикладных информационных моделей с использованием современных языков программирования. Формирование у студентов объектно-ориентированного мышления, изучение объектно-ориентированной методологии программирования, ознакомление студентов с основными понятиями и приемами программирования на языках высокого уровня, с интегрированными средами разработки; выработка способности самостоятельно формализовать задачу, разрабатывать структуру программы, тестировать программу; ознакомление с техническими решениями, используемыми для написания программных комплексов; выработка практических навыков написания на языках C, C++, разработки пользовательского интерфейса. Формирование компьютерной грамотности и подготовка студентов к использованию компьютеров и современной технологии программирования в качестве инструмента для решения практических задач в своей предметной области. А также содействовать фундаментализации образования, формированию научного мировоззрения и развитию системного мышления. Предлагаемый курс также способствует расширению кругозора и воспитанию программистской культуры. Воспитание у студентов программистской культуры включает в себя четкое представление роли языков программирования высокого уровня в современной социально-экономической деятельности, а также получение необходимых практических навыков прикладного программирования.

Задачей преподавания дисциплины является формирование у студентов профессиональных компетенций, связанных с использованием теоретических и практических знаний в области прикладного программирования на языке высокого уровня. Основные задачи дисциплины: подготовка к осознанному использованию как языков программирования, так и методов программирования.

3. Место дисциплины (модуля) в структуре ООП

Дисциплина входит в блок Дисциплины по выбору и формирует компетенцию **ПК-12**.

Дисциплина читается на 4 курсе (8 семестр), заканчивается зачетом. Для освоения дисциплины студент должен владеть современными методами и средствами информационных технологий. Необходимы знания, умения и компетенции, сформированные в процессе обучения по дисциплинам Информатика, Принципы алгоритмизации, Языки программирования, Методы программирования.

Данная дисциплина является одним из базовых курсов для профессиональной подготовки специалистов в области разработки программного обеспечения сложных программных систем для различных проблемных областей. Она является базовой для изучения дисциплин Web–дизайн, Интернет-программирование, Программное обеспечение компьютерных систем. Знания и практические навыки, полученные из курса, используются студентами при изучении научных дисциплин, при прохождении производственной и преддипломной практики, а также при разработке курсовых и дипломных работ.

4. Объем дисциплины (или модуля):

3 __ зачетных единиц, **108** __ академических часов, в том числе **контактная работа:** лекции __15__ часов, практические занятия __15__ часов, **самостоятельная работа:** __78__ часов.

5. Перечень планируемых результатов обучения по дисциплине (или модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Планируемые результаты освоения образовательной программы (формируемые компетенции)	Планируемые результаты обучения по дисциплине (или модулю)
<p>ПК-12 – способностью проводить инструментальный мониторинг защищенности компьютерных систем</p>	<p>Владеть:</p> <ul style="list-style-type: none"> • профессиональной терминологией в области ИТ; • средствами и средой программирования, современными технологиями программирования, методами настройки и отладки. • принципами построения многомодульных программ и программных комплексов. <p>Уметь:</p> <ul style="list-style-type: none"> • формализовать поставленную задачу; • использовать языки и системы программирования для решения профессиональных задач, • использовать библиотеки алгоритмов и подпрограмм для графических компонентов, проектировать пользовательский интерфейс прикладных программ; • тестировать и отлаживать программы; <p>Знать:</p> <ul style="list-style-type: none"> • Принципы, базовые концепции технологий программирования, основные этапы и принципы создания программного продукта. Жизненный цикл

	программного средства <ul style="list-style-type: none"> • основы разработки алгоритмов и программ с использованием высокоуровневых языков программирования; • основы построения пользовательских интерфейсов: консольных и графических; • методы повышения надежности программирования с применением объектного подхода;
--	--

6. Форма промежуточной аттестации зачёт

7. Язык преподавания русский.

II. Содержание дисциплины (или модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

1. Для студентов очной формы обучения

Учебная программа – наименование разделов и тем	Всего (час.)	Контактная работа (час.)		Самостоятельная работа (час.)
		Лекции	Практические (лабораторные) занятия	
Тема 1. Технологии разработки прикладного программного обеспечения	7	1	1	5
Тема 2. Объектно-ориентированное программирование (ООП) и основы проектирования программного обеспечения.	7	1	1	5
Тема 3. Методы и механизмы разработки объектно-ориентированных программ.	13	2	2	9
Тема 4. Применение ООП в разработке прикладных программ. Применение объектно-ориентированного анализа и проектирования. Компонентный подход. Повторное использование кода. Совместная разработка элементов приложений.	13	2	2	9
Тема 5. Универсальный язык моделирования UML.	26	2	4	20
Тема 6. Пользовательский интерфейс прикладных программ	42	7	5	30
ИТОГО	108	15	15	78

Учебная программа

- 1. Тема 1. Технологии разработки прикладного программного обеспечения**
 - 1.1. Технологии прикладного программирования: цели, задачи и основные принципы и инструменты.
 - 1.2. Сложность программного обеспечения Основные причины сложности. Сложность управления процессом разработки. Гибкость программного обеспечения. Признаки сложных систем.
 - 1.3. Алгоритмическая и объектно-ориентированная декомпозиция.
 - 1.4. Объектно-ориентированное программирование. История появления. Основные идеи. Преимущества ООП.
 - 1.5. Базовые принципы ООП. Концепции ООП: Объекты, абстракция, инкапсуляция, полиморфизм, наследование, агрегирование.
- 2. Тема 2. Объектно-ориентированное программирование (ООП) и основы проектирования программного обеспечения.**
 - 2.1. Базовые конструкции объектно-ориентированных программ: классы и объекты.
 - 2.2. Компоненты класса. Элементы класса: поля и методы. Управление видимостью элементов класса.
 - 2.3. Поведение объекта: операции (модификатор, селектор, конструктор, деструктор).
 - 2.4. Инициализация и разрушение объекта. Инициализация объектов класса: конструкторы, инициализаторы конструктора. Конструкторы и деструкторы классов. Последовательность вызова конструкторов и деструкторов.
 - 2.5. Отношения между объектами: связи и агрегация.
 - 2.6. Статические поля и методы класса.
- 3. Тема 3. Методы и механизмы разработки объектно-ориентированных программ.**
 - 3.1. Проектирование программного обеспечения с использованием механизма наследования. Понятие наследования. Базовый класс и производный классы.
 - 3.2. Открытые, защищенные и закрытые базовые классы.
 - 3.3. Конструкторы и деструкторы в производных классах.
 - 3.4. Проектирование программного обеспечения с помощью наследования, построение иерархии классов, доступ к объектам иерархии.
 - 3.5. Композиция и наследование. Множественное наследование.
- 4. Тема 4. Применение ООП в разработке прикладных программ. Применение объектно-ориентированного анализа и проектирования. Компонентный подход. Повторное использование кода. Совместная разработка элементов приложений.**
 - 4.1. Полиморфизм, его основные проявления, механизмы использования.
 - 4.2. Понятие раннего и позднего связывания.
 - 4.3. Использование виртуального механизма для реализации принципа полиморфизма.
- 5. Тема 5. Универсальный язык моделирования UML.**
 - 5.1. Моделирование взаимодействия между объектами.

- 5.2. Диаграммы классов, диаграммы последовательностей, диаграммы кооперации, диаграммы деятельности.
- 5.3. Порождающие шаблоны проектирования. Назначение порождающих шаблонов.
- 5.4. Структурные шаблоны проектирования. Назначение структурных шаблонов проектирования. Структурные шаблоны уровня класса и уровня объекта.
- 5.5. Шаблоны поведения. Назначение шаблонов поведения.

6. Тема 6. Пользовательский интерфейс прикладных программ. Процесс разработки графического интерфейса.

- 6.1. Проектирование графического интерфейса пользователя.
- 6.2. Понятие элементов управления, обработчиков событий, виды событий, классы элементов управления.
- 6.3. Средства визуального конструирования графического интерфейса.
- 6.4. Программное средство fluid из состава инструментария FLTK.
- 6.5. Иерархия классов и основные методы для работы с элементами управления.

III. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (или модулю)

Планы практических (семинарских) занятий и методические рекомендации к ним.

Методические рекомендации по организации самостоятельной работы студентов.

Тематика рефератов и рекомендации по их оформлению

Тестовые вопросы для самоконтроля

IV. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине (или модулю)

1. Типовые контрольные задания для проверки уровня сформированности компетенции

ПК-12 – способностью проводить инструментальный мониторинг защищенности компьютерных систем.

Рассматривается трехкомпонентной структура компетенции: знать, уметь, владеть.

При этом под указанными категориями понимается:

- «знать» – воспроизводить и объяснять учебный материал с требуемой степенью научной точности и полноты;
- «уметь» – решать типичные задачи на основе воспроизведения стандартных алгоритмов решения;
- «владеть» – решать усложненные задачи на основе приобретенных знаний, умений и навыков, в нетипичных ситуациях

Этап формирования	Типовые контрольные задания	Показатели и критерии оценивания компетенции, шкала
-------------------	-----------------------------	---

компетенции, в котором участвует дисциплина	для оценки знаний, умений, навыков (2-3 примера)	оценивания
владеть	Составить описание класса для представления комплексных чисел.	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части описания из-за логической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
	Описать класс, реализующий стек.	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за логической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
уметь	Объясните принцип инкапсуляции.	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов

	Дана целочисленная прямоугольная матрица. Определить максимальное из чисел, встречающихся в заданной матрице более одного раза.	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за логической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
знать	Написание реферата	<ul style="list-style-type: none"> • Отражение в плане ключевых аспектов темы – 2 балла; • Фрагментарное отражение ключевых аспектов темы – 1 балл; • верно оформлены ссылки на используемую литературу – 1 балл • соблюдены требования к объёму реферата – 1 балл.
	Пояснить назначение абстрактных классов, виртуальных функций.	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов
владеть	Прокомментировать по программе использование основных принципов ООП.	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов

	<p>Проследить правильную организацию исключений</p>	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов
<p>уметь</p>	<p>Реализуйте класс String для работы со строками символов</p>	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за алгоритмической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
<p>знать</p>	<p>Понятие раннего и позднего связывания.</p>	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов
	<p>Структурные диаграммы и диаграммы поведения UML</p>	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов

владеть	Смоделировать битву двух противников.	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части модели из-за логической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
	Привести пример применения шаблона поведения «Стратегия».	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за алгоритмической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
уметь	применение шаблона проектирования «Одиночка».	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за логической ошибки – 1 балл • Решение не дано ИЛИ дано неверное решение – 0 баллов
	Демонстрация умения использовать язык программирования; Спроектировать и реализовать игру «Тетрис»	<ul style="list-style-type: none"> • Имеется полное верное решение, включающее правильный ответ – 3 балла • Дано верное решение, но в решении имеются неверные записи, не отделенные от решения – 2 балла • Имеется верное решение части программы из-за алгоритмической ошибки – 1 балл • Решение не дано ИЛИ

		дано неверное решение – 0 баллов
знать	Система каталогизации шаблонов проектирования	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов
	Основные причины сложности программного обеспечения.	<ul style="list-style-type: none"> • Факты и примеры в полном объеме обосновывают выводы – 2 балла • Допущена фактическая ошибка, не приведшая к существенному искажению смысла – 1 балл • Допущены фактические и логические ошибки, свидетельствующие о непонимании темы – 0 баллов

При оценивании результатов освоения дисциплины применяется «рейтинговая» технология (балльно-накопительная) система. Оценка уровня сформированности компетенций осуществляется в процессе следующих форм контроля:

1) **слеящегоо** (проводится оценка выполнения студентами заданий в ходе аудиторных занятий). Дает возможность квалифицировать степень сформированности знаний, умений, навыков, а также их глубину и прочность. Его задача - регулярное управление учебной деятельностью студентов и ее корректировка. Он позволяет получать первичную информацию о ходе и качестве усвоения учебного материала, а также стимулировать регулярную, напряженную и целенаправленную работу студентов. Данный контроль позволяет вовремя выявить пробелы в знаниях и оказать им помощь в усвоении

программного материала. Данными формами контроля являются: ответы с места и у доски, проверка работ выполненных в тетради.

- 2) **текущего** (оценивается работа студентов вне аудиторных занятий). Текущими формами контроля являются: проверка выполнения практических работ, ответы у доски, рефераты, доклады, проверка самостоятельной работы студентов.
- 3) **промежуточного** (рейтинговые точки) позволяет определять качество изучения студентами учебного материала по разделам и темам. Контроль проводится два раза в семестр. С помощью периодического контроля обобщаются и усваиваются целые темы и разделы, выявляются взаимосвязи с другими разделами, предметами. Контроль охватывает студентов и всей группы и проводится в виде теста, письменных практических работ.
- 4) **итогового** (зачёт). Максимальная сумма рейтинговых баллов по дисциплине составляет 100 баллов. Студенту, набравшему 50 баллов и выше по итогам работы в семестре, в экзаменационной ведомости и зачетной книжке выставляется оценка «зачтено». Студент, набравший от 20 до 49 баллов включительно, сдаёт зачет в последнюю неделю семестра по данной дисциплине. Баллы, полученные на зачете проставляются в ведомости. Студенту, набравшему меньше 20 баллов, в экзаменационной ведомости выставляется оценка «незачтено». Данному студенту разрешается передача зачета по направлению деканата на последней неделе семестра.

Вид учебной работы, за которую ставятся баллы	Баллы
оценка выполнения студентами заданий в ходе аудиторных занятий, устный опрос	0-10
реферат	0-2
Практическая работа 1	0-3
Практическая работа 2	0-5
Практическая работа 3	0-10
Практическая работа 4	0-10
Контрольный тест к модулю 1	0-10
Контрольный тест к модулю 2	0-10
Контрольная работа 1	0-10
Контрольная работа 2	0-10
Итоговый контрольный тест	0-20
Всего	0-100

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующие этапы формирования компетенций в процессе освоения образовательной программы.

Для оценки уровня теоретических и практических знаний используется тест или контрольный письменный опрос. Перечень некоторых вопросов теста и практических заданий представлен ниже.

Тест 1.




1. Какое определение паттернов проектирования (design patterns) правильно?

- множество схем разбиения классов на составные части со спецификацией их ответственности и отношений между ними
- специальные схемы для организации программного кода в модулях реализации системы
- специальные схемы для уточнения структуры подсистем или компонентов программной системы и отношений между ними

2. Какое определение конкретного класса (concrete class) правильно?

- класс, который имеет заданные типы атрибутов и операций
- класс, который содержит реализацию своих операций
- класс, на основе которого могут быть непосредственно созданы экземпляры или объекты

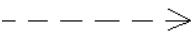


Какой графический символ служит для изображения примечания (note) в языке UML?

- 
- 
- 

3. Какие из перечисленных диаграмм относятся к каноническим в языке UML?

- диаграмма артефактов
- диаграмма деятельности
- диаграмма компонентов
- диаграмма экземпляров классов
- диаграмма IDEF3

4. Как изображается отношение агрегации (aggregation) на диаграмме классов?

- 
- 
- 

Задание 1

Вам даны классы BinaryOperation (бинарный оператор) и Number (число), которые наследуются от базового класса Expression (выражение). Ваша задача реализовать базовый класс Expression так, чтобы не было утечек памяти. Кроме этого подумайте, какие методы стоит сделать виртуальными.

Решение 2 строки кода в первом классе

```
#include <cassert> // assert
```

```
struct Expression{  
    //////////////// (это решение) virtual double evaluate() const = 0;  
    //////////////// (это решение) virtual ~Expression(){ }  
};
```

```
struct Number : Expression {  
    Number(double value)  
        : value_(value)  
    {}  
  
    double value() const { return value_; }
```

```

    double evaluate() const { return value_; }

private:
    double value_;
};

struct BinaryOperation : Expression {
    enum {
        PLUS = '+',
        MINUS = '-',
        DIV = '/',
        MUL = '*'
    };

    BinaryOperation(Expression const *left, int op, Expression const *right)
        : left_(left), op_(op), right_(right)
    { assert(left_ && right_); }

    ~BinaryOperation() {
        delete left_;
        delete right_;
    }

    Expression const *left() const { return left_; }

    Expression const *right() const { return right_; }

    int operation() const { return op_; }

    double evaluate() const {
        double left = left_->evaluate();
        double right = right_->evaluate();
        switch (op_) {
            case PLUS: return left + right;
            case MINUS: return left - right;
            case DIV: return left / right;
            case MUL: return left * right;
        }
        assert(0);
        return 0.0;
    }

private:
    Expression const *left_;
    Expression const *right_;
    int op_;
};

```

Задание 2

Добавьте к иерархии из предыдущего упражнения класс `FunctionCall`. `FunctionCall` должен представлять вызов одной из двух predefined математических функций: `sqrt` — извлечение квадратного корня и `abs` — вычисление модуля числа. Функция идентифицируется строкой, переданной в качестве параметра в конструктор.

Решение - добавлено: деструктор, и функция double evaluate() const

```
#include <string> // std::string
#include <cassert>
#include <cmath> // sqrt и fabs

// эти классы определять заново не нужно
struct Expression;
struct BinaryOperation;
struct Number;

struct FunctionCall : Expression{
    /**
     * @name имя функции, возможные варианты
     * "sqrt" и "abs".
     *
     * Объекты, std::string можно
     * сравнивать с C-строками используя
     * обычный синтаксис ==.
     *
     * @arg выражение аргумент функции
     */
    FunctionCall(std::string const &name, Expression const *arg) : name_(name), arg_(arg) {
        assert(arg_);
        assert(name_ == "sqrt" || name_ == "abs");
    }

    // реализуйте оставшиеся методы из
    // интерфейса Expression и не забудьте
    // удалить arg_, как это сделано в классе
    // BinaryOperation

    double evaluate() const {
        if (name_ == "sqrt")
            return sqrt(arg_->evaluate());
        else if ( name_ == "abs")
            return fabs(arg_->evaluate());
        assert(0);
        return 0.0;
    }

    ~FunctionCall (){ delete arg_; }

    std::string const & name() const { return name_; }

    Expression const *arg() const { return arg_; }

    // and here

private:
    std::string const name_;
    Expression const *arg_;
};
```

Задание 3

1. Использование STL.

Целью работы является ознакомление и освоение стандартной библиотеки языка C++, в части использования контейнерных классов и методов для работы с его элементами. Использование контейнеров позволяет значительно повысить надежность программ, их переносимость и универсальность.

Методические указания к работе

1. Контейнер — это объект, содержащий набор других объектов, организованный определенным образом. Контейнеры предназначены для управления коллекциями объектов определенного типа. Существование разных контейнеров отражает различие между требованиями к коллекциям в программах.
2. Один и тот же вид контейнера можно использовать для хранения и работы с объектами различных типов. Такая возможность реализуется с помощью шаблонов классов.
3. Основные требования, которые должны выполняться контейнером: - контейнеры должны поддерживать семантику значений вместо ссылочной семантики. Это означает, что при вставке элемента контейнер должен создавать его внутреннюю копию, вместо того чтобы сохранять ссылку на внешний объект; -элементы в контейнере должны располагаться в определенном порядке. Это означает, что при повторном переборе элементов контейнера с применением итератора порядок перебора элементов должен остаться прежним. Итераторы представляют собой основной интерфейс для работы алгоритмов STL.

Разработать пример использования одного из классов (по своему выбору) стандартной библиотеки шаблонов.

Продемонстрировать создание, наполнение, извлечение из набора.

Контрольные вопросы:

1. Что представляют собой контейнеры, зачем они нужны?
2. Что представляют собой последовательные и ассоциативные контейнеры?
3. Назовите для контейнеров общие возможности, унифицированные типы и общие операции и методы.
4. Для чего предназначены итераторы?
5. Какие операции допустимы для любого типа итератора?

2. Классы. Наследование. Создание простых иерархий.

Цель работы – освоение и закрепление основных навыков объектно-ориентированного программирования, связанных с проектированием спецификаций класса, его структуры и операций интерфейса с учетом свойства инкапсуляции.

Спроектировать и реализовать иерархию классов для игры в шахматы.

В качестве варианта использования рассмотреть задачу анализа ситуации на поле. В задаче участвует несколько черных фигур (любого вида) и одна белая – король. Провести анализ положения белого короля в условии хода белых фигур - шах, пат, мат или белому королю ничего не угрожает.

В иерархии классов реализовать:

- проверку попадания фигуры на доску;
- проверку установки фигуры на свободную клетку;
- отслеживание количества экземпляров каждого вида фигур.

Методические указания к работе

Разработанная иерархия должна допускать использование:

```
int main() {
    TKing bKing(black, 'h', 5);
    THorse bHorse1(black, 'c', 2);
    TKing bKing2(black, 'a', 1); // эта строчка работать не должна, потому что двух
                                // одинаковых королей быть не может
    THorse bHorse2(black, 'c', 2); // эта строчка работать не должна, потому что клетка
                                // c2 уже занята
    THorse bHorse3(black, 'z', 20); // эта строчка работать не должна из-за
                                // неправильных координат

    TFigure *figures[32];
    figures[0]=&bKing;
    figures[1]=&bHorse;

    /*for (int i = 0; i < 32; i++)
        figures[i]->mapStep();*/

    TKing wKing(white, 'a', 4);
    cout << endl << wKing.Situation();

    return 0;
}
```

В тестовой программе информацию о черных фигурах считывать из файла, координаты белого короля получать с клавиатуры.

Для удобства проверки правильности работы программы реализовать метод вывода доски на экран.

Контрольные точки

1. Прокомментировать по программе использование основных принципов ООП.
2. Пояснить назначение абстрактных классов, виртуальных функций.
3. Проследить правильную организацию исключений.

3. Классы. Наследование. Создание простых иерархий.

Целью работы является создание производного класса от базового класса.

Некий город представляет собой квадрат со стороной 1 км, в котором улицы параллельны сторонам и идут через каждый 100 м (в том числе и по границам города). Написать класс автомобиль, моделирующий поездку на автомобиле по такому городу. Он должен иметь конструктор по координатам и направлению, конструктор по умолчанию (устанавливающий автомобиль на центральном перекрестке направленным на север), а также методы повернуть на 90° направо, повернуть налево, проехать указанное число кварталов прямо (при этом автомобиль не должен выезжать за пределы города). Написать ещё один класс, моделирующий поездку автобуса (кроме вышеперечисленного, у него должны быть методы “войти” и “выйти” с указанием числа пассажиров, и он должен отслеживать плату за проезд из расчета один рубль на поездку одного пассажира на один квартал; соответственно

должен быть метод, возвращающий количество собранных денег). Разработать перечень команд, с помощью которых пользователь управляет движением автомобиля и автобуса.

Например,

s x y d (или set x y d) – установить автомобиль на позицию с координатами x и y и с направлением d (значениями направления, например, могут быть просто числа 1, 2, 3, 4, означающие север, юг, восток, запад соответственно);

l (left) – повернуть налево;

r (right) – повернуть направо;

f x (forward x) – проехать вперед x кварталов;

w (where) – вывести текущие координаты и направление автомобиля/автобуса;

i n (in n) – входят n человек;

o n (out n) – выходят n человек;

m (money) – вывести количество собранных денег;

exit – завершить работу.

4. Классы. Наследование и полиморфизм.

Целью работы является освоение такого важного аспекта языка C++ как виртуальные функции, с помощью которых поддерживается динамический полиморфизм.

Разработайте небольшую игру, используя принципы наследования для задания игровых объектов, таких как «Монстры» и «Оружие». Реализацию программы и ее архитектуру опишите в отчете.

Методические указания к работе

Наследование и полиморфизм при наследовании являются одним из основных способов задания сложных программных архитектур, построенных на взаимодействии различных объектов. Для иллюстрации рассмотрим следующий пример.

Пусть в игре существуют монстры. Каждый монстр обладает определенным количеством жизней, урона при ударе и, возможно, именем. В этом случае для описания всех монстров нам достаточно будет одного класса `Monster`, содержащего соответствующие поля для хранения жизней, урона, имени и т.п.

Предположим теперь, что для некоторых видов монстров существуют особенности ведения боя. Например:

- монстр типа 1 удваивает наносимый урон, если количество его жизней становится менее 10 единиц;
- монстр типа 2 с вероятностью 50% наносит два удара за ход;
- монстр типа 3 наносит удар через раз, но с утроенной силой.

В классическом процедурном подходе такие задачи разрешались бы при помощи инструкций ветвления (**if** или **switch**), загромождая и утяжеляя код.

Подход с наследованием позволяет выделить весь общий код в базовую реализацию класса `Monster`, вынеся все особенности поведения монстров разных типов в наследные классы `Monster1`, `Monster2` и т. п.

Более того, классу игрока (Player) нет никакой надобности знать, с экземпляром какого именно класса он в данный момент сражается — экземпляры всех унаследованных классов Monster1 и т.д. по полиморфизму будут являться также экземплярами класса Monster, с которыми и будет сражаться игрок.

Предположим далее, что и у игрока и у монстра бывают различные виды оружия. В общем случае оружие представляется классом Weapon. Но у этого класса могут быть разные наследники, каждый со своими характеристиками скорострельности, урона, промахов и т.п.

Разработайте программу, симулирующую в том или ином виде взаимодействие игрока и монстров, вооруженных различными видами оружия. Структуру вашей программы и описание игровых механик представьте в виде отчета.

1. Полиморфизм в языке C++ поддерживается двумя способами: посредством перегрузки операций и функций при компиляции; вовремя выполнения программы — с помощью динамического полиморфизма. Лабораторная работа должна использовать оба эти способа.

2. Основой виртуальных функций и динамического полиморфизма являются указатели на производные классы.

3. Арифметика указателей связана с типом данных (т.е. с классом), который задан при объявлении указателя. Таким образом, если указатель базового класса указывает на объект производного класса, а затем инкрементируется, то он уже не будет указывать на следующий объект производного класса.

4. По существу, виртуальная функция реализует идею «один интерфейс, множество методов», которая лежит в основе полиморфизма.

5. Тип адресуемого через указатель базового класса объекта определяет вызов той или иной версии подменяемой виртуальной функции.

6. Для выполнения лабораторной работы рекомендуется разработать класс IOFile, производный от класса fstream. Класс должен обеспечивать использование перегруженных операций ввода (>>) и вывода (<<) для стандартных типов.

Задание 4

4. Паттерны проектирования.

Спроектировать каркас для построения ролевой игры. В игре участвуют несколько цивилизаций, как минимум 3 — первая, вторая и третья. В каждой представлены индивидуумы нескольких видов: воин, рабочий, аристократ и пр.

Каждая раса обладает финансовым запасом и некоторым набором территорий с первоначально обязательно размещенными на них объектами типов: леса, поля, жилища и производящие объекты.

В тестирующей программе продемонстрировать создание различных рас, отрядов индивидуумов, инициализацию территорий и изменение состава объектов игры.

Методические указания

Использовать порождающие паттерны.

5. Работа с графическим интерфейсом пользователя. Обработка событий.

Цель работы. Знакомство с элементами разработки графического интерфейса пользователя. Создание приложения из нескольких файлов.

Содержание работы. В данной работе рассматриваются программы, демонстрирующие основы работы с графическим интерфейсом пользователя и обработкой событий компонентов GUI.

- Разработать графический интерфейс программы «научный калькулятор» с поддержкой специальных функций. Предусмотреть вывод графиков функций.
- Создать приложение- калькулятор. Разработать грамматику вычисления выражений с произвольным уровнем скобок и использованием переменных и функций. Реализовать калькулятор с использованием данной грамматики.

Методические указания

Средства визуального конструирования графического интерфейса.

Программное средство fluid (**F**ast **L**ight **U**ser-**I**nterface **D**esigner) из состава инструментария FLTK.

В оболочке Visual C++ необходимо указать компилятору место нахождения заголовочных файлов FLTK. (в закладке C/C++ ->Project -> Settings). Добавить библиотеки FLTK и WinSock (WSOCK32.LIB) в установках "Link".

Правильный способ подключения заголовочных файлов FLTK:

```
#include <FL/Fl_xyz.H>
```

FLUID – это, графический редактор пользовательского интерфейса, использующийся для получения исходного кода на языке описания интерфейса FLTK. FLUID редактирует и сохраняет файлы с суффиксом .fl. Для запуска FLUID под WIN32 используйте двойной щелчок на файле FLUID.exe.

Вопросы для подготовки к промежуточной аттестации:

1. Что такое класс, объекты?
2. Дайте определение атрибуту (свойству) и поведению объекта.
3. Опишите методы и атрибуты класса Карта, представляющего собой карту в карточных играх.
4. Дайте определение наследованию.
5. Опишите преимущества наследования.
6. Какой из этих классов является базовым: Транспортное средство или Микроавтобус?
7. Какие из следующих классов связаны отношением наследования: классы Двигатель и Дизель или классы Двигатель и Автомобиль?
8. Что такое конструктор и деструктор?
9. Дайте определение инкапсуляции.
10. Как можно обратиться к членам объекта?
11. Дайте определение полиморфизма.
12. Дайте определение перегрузки.

13. Дайте определение связывания.
14. Дайте определение полиморфизма времени выполнения.
15. Что такое виртуальная функция?
16. Как описывается виртуальный метод в C++?
17. Дайте определение простого и множественного наследования.
18. Дайте определение абстракции.
19. В чем заключается аналогия между понятиями, используемыми при объектно-ориентированном подходе к программированию, и существительными и глаголами?
20. Какой синтаксис используется в C++ для объявления класса, производного от базового?
21. Полиморфизм времени выполнения (динамический полиморфизм).
22. Виртуальные функции. Абстрактные классы. Иерархии классов и абстрактные классы.
23. Шаблоны. Определение шаблонов функций.
24. Полиморфизм времени компиляции (параметрический полиморфизм).
25. Основные концепции – контейнеры, итераторы и алгоритмы.
26. Обзор алгоритмов стандартной библиотеки. Итераторы и распределители памяти.

Список вопросов к зачету

1. Программные продукты и их основные характеристики. Понятие программного обеспечения. Характеристики качество программного обеспечения.
2. Проектирование программного продукта. Проектирование программы. Основные этапы проектирования. Реализация программы.
3. Структура программного продукта.
4. Проектирование интерфейса пользователя.
5. Методы повышения надёжности программ.
6. Этапы создания программных продуктов.
7. В чем состоит концептуальное отличие структурного программирования от объектно-ориентированного программирования?
8. Перечислите этапы объектной декомпозиции.
9. Какие события формы Вы знаете?
10. В каком порядке выполняются события клавиатуры: OnKeyPress, OnKeyUp, OnKeyDown?
11. Какие события происходят при создании формы и в какой последовательности?
12. Какие события происходят при закрытии формы и в какой последовательности?
13. Охарактеризуйте жизненный цикл программного обеспечения.
14. Дайте характеристику каждого этапа разработки программного продукта.
15. STL - Контейнеры STL (вектор, двусвязный список, множество, карта (отображение), двусторонняя очередь) Итераторы. Алгоритмы STL.
16. Основные концепции ООП. Понятия класса и объекта. Абстракция. Инкапсуляция. Наследование. Полиморфизм. Методы, данные и свойства. Ограничение доступа к полям классам.
17. Классы. Методы и поля (данные) классов. Объявление класса в C++.

18. Разграничение доступа к полям и методам класса (спецификаторы доступа). Интерфейс и реализация класса.
19. Классы: функции-члены, конструкторы и деструкторы. Списки инициализации.
20. Конструкторы и деструкторы. Перегрузка конструкторов. (Конструктор по умолчанию. Конструктор копирования. Конструктор с аргументами)
21. Классы: Указатель `this`. Друзья класса. Дружественные классы и функции. Статические данные и методы. Особенности. Область применения. Пример.
22. Вложенные классы. Композиция.
23. Шаблоны класса: определение и инстанцирование.
24. Классы: перегрузка функций. перегрузка операторов.
25. Перегрузка бинарных операторов
26. Перегрузка операторов отношения и логических операторов
27. Перегрузка унарных операторов Перегрузка оператора присваивания и индекса массива
28. Классы: Принцип наследования в ООП. Варианты наследования. Публичное наследование. Защищенное наследование. Закрытое наследование. Иерархия классов
29. Классы: виртуальные функции и абстрактные классы.
30. Обобщенное программирование. Шаблоны функций. Шаблоны операторов. Шаблоны классов. Параметры шаблонов, не являющиеся типами.
31. Полиморфизм, его основные проявления, механизмы использования.
32. Понятие раннего и позднего связывания.
33. Использование виртуального механизма для реализации принципа полиморфизма.
34. Универсальный язык моделирования UML.
35. Диаграммы классов
36. Диаграммы последовательностей
37. Диаграммы деятельности.
38. Шаблоны проектирования. Основные термины и понятия.
39. Система каталогизации шаблонов проектирования.
40. Порождающие шаблоны проектирования. Назначение порождающих шаблонов.
41. Структурные шаблоны проектирования. Назначение структурных шаблонов проектирования.
42. Шаблоны поведения. Назначение шаблонов поведения.
43. Проектирование графического пользовательского интерфейса
44. Современные способы построения интерфейсов
45. Методы и средства разработки пользовательского интерфейса: современное состояние.
46. Примеры разработки пользовательского графического интерфейса в сфере информационных технологий.

V. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (или модуля)

а) Основная литература:

1. Агафонов, Е.Д. Прикладное программирование: учебное пособие / Е.Д. Агафонов, Г.В. Ващенко; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск: Сибирский федеральный университет, 2015. - 112 с. : табл., граф., ил. - Библиогр. в кн. - ISBN 978-5-7638-3165-8 ; То же [Электронный ресурс]. – Режим доступа: <http://biblioclub.ru/index.php?page=book&id=435640>
2. Снетков В.М. Практикум прикладного программирования на С# в среде VS.NET 2008 [Электронный ресурс]/ В.М. Снетков.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 1691 с.— Режим доступа: <http://www.iprbookshop.ru/62823.html>
3. Иванов В.Б. Прикладное программирование на С/С++. С нуля до мультимедийных и сетевых приложений [Электронный ресурс]/ В.Б. Иванов.— Электрон. текстовые данные.— М.: СОЛОН-ПРЕСС, 2011.— 240 с.— Режим доступа: <http://www.iprbookshop.ru/65139.html>

б) Дополнительная литература:

1. Лисицин Д.В. Объектно-ориентированное программирование [Электронный ресурс]: конспект лекций/ Лисицин Д.В.— Электрон. текстовые данные.— Новосибирск: Новосибирский государственный технический университет, 2010.— 88 с.— Режим доступа: <http://www.iprbookshop.ru/44970.html>
2. Визуальное программирование на основе библиотеки MFC [Электронный ресурс]: методические указания к лабораторным работам по курсу «Визуальное программирование», 2016.— 57 с.—Режим доступа: <http://www.iprbookshop.ru/28324.html>

VI. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (или модуля)

1. ЭБС Лань <https://e.lanbook.com/> Договор № 4-е/23 от 02.08.2023 г.
2. ЭБС Znanium.com <https://znanium.com/> Договор № 1106 эбс от 02.08.2023 г.
3. ЭБС Университетская библиотека online <https://biblioclub.ru> Договор № 02-06/2023 от 02.08.2023 г.
4. ЭБС ЮРАЙТ <https://urait.ru/> Договор № 5-е/23 от 02.08.2023 г.
5. ЭБС IPR SMART <https://www.iprbookshop.ru/> Договор № 3-е/23К от 02.08.2023 г.

VII. Методические указания для обучающихся по освоению дисциплины (или модуля)

Материал дисциплины распределен по главным разделам (темам). В результате изучения дисциплины у студентов должно сформироваться научное представление о системах программирования. Необходимо выработать системный подход к пониманию процессов разработки компьютерных приложений. В процессе обучения студенты, наряду с текстами лекций и учебными пособиями, должны пользоваться

дополнительными научными изданиями, академическими периодическими изданиями. После каждой лекционной темы рекомендуется проработать вопросы для повторения и самоконтроля. В аспекте самостоятельной работы рекомендуется составлять конспект с наиболее важными методами и приемами создания приложений. Рекомендуется использовать справочники и руководства.

Для успешного освоения дисциплины важно соблюдать следующие рекомендации: На первой лекции важно обратить внимание на конкретные требования к прохождению и сдаче курса. Активная работа на занятиях, выполнение творческих заданий сформирует у Вас дополнительное положительное представление как об активном участнике познавательного процесса. На данном курсе практические занятия являются самым важным компонентом обучающего процесса. На занятиях будет представлен необходимый теоретический материал по темам и представлены практические задания для решения на занятиях в аудитории под руководством преподавателя и самостоятельно. Многие задачи являются стандартными и имеют уже готовые шаблоны (алгоритмы) решения, тем не менее, для получения большего познавательного и учебного эффекта, настоятельно рекомендуется написание собственного оригинального кода.

Самостоятельная работа студентов в рамках данной дисциплины в основном состоит в подготовке к практическим занятиям и написании алгоритмов и программ, в работе с разными источниками. Освоению учебного материала большую помощь окажет личный творческий подход, связанный с дополнительным просмотром материала по отдельным темам в библиотеках и системе «Интернет». Самостоятельная работа является необходимой на всех стадиях и при всех формах изучения предмета. Важно помнить: без самостоятельной работы невозможно серьезное освоение любого курса. Надо быть готовым к тому, что по времени, затраченному на дисциплину, самостоятельная работа будет преобладать над иными видами работы. Важно продумать стиль фиксации нового и важного материала. Рекомендуется немедленно обсуждать любые возникшие в процессе обучения вопросы, проблемы и неясности с преподавателем, не откладывая это обсуждение до контрольной точки. Проконсультироваться с преподавателем можно во время и после практических занятий, во время консультаций, а также по электронной почте.

Самостоятельная работа обучающихся направлена на освоение учебного материала и развитие практических умений. Самостоятельная работа включает следующие виды самостоятельной работы студентов: работа с рекомендованной литературой и документацией; выполнение практических заданий; подготовка к контрольным тестам и итоговому зачёту.

Список практических заданий

1. Разработать абстрактный тип данных «геометрические фигуры на плоскости», включающие элементарные структуры, операции (перенос, поворот) и тесты (пересекаются ли, находятся ли внутри или снаружи).
2. Разработать абстрактный тип «матрица», включающий структурный тип данных, операции сложения, умножения, определители, обращение матриц.
3. Разработать графический интерфейс программы «научный калькулятор» с поддержкой специальных функций. Предусмотреть вывод графиков функций.

4. Создать приложение- калькулятор. Разработать грамматику вычисления выражений с произвольным уровнем скобок и использованием переменных и функций. Реализовать консольный калькулятор с использованием данной грамматики. Присоединить ранее разработанный модуль интерфейса из предыдущей темы.
5. Написать программу для решения логической задачи (варианты: переправа через реку, поиск пути в лабиринте)
6. Написать программу имитационного моделирования экосистемы с несколькими пищевыми цепочками (например, волк+зайцы+травы).
7. Написать программу с использованием методов динамического программирования (прохождение лабиринта, поиск критического пути на графе).
8. Реализовать игру «Тетрис» в текстовом или в графическом режиме.
9. С помощью ООП разработать возможную реализацию некоторого компьютерного мира. Обитатели такого мира могут иметь различную форму, быть подвижными и неподвижными, быстрыми и медленными, могут размножаться, скрещиваться, нападать и защищаться и т.д. (выполнять могут до трех человек в группе)
10. Разработать объект «Меню», управляемый с помощью кнопок.
11. Смоделировать битву двух противников.
12. Описать объект строка, инкапсулирующий методы для работы со строками.
13. Создать простейший векторный графический редактор для рисования линий, различных геометрических фигур, заливки и т.д.
14. Создать приложение, показывающее картинки из пяти различных коллекций (Части света: Азия, Африка, Европа, Америка, Австралия), посвященных городам мира. Приложение должно работать так: выбираем часть света, появляется список соответствующих городов. При выборе города из этого списка появляется его фотография.
15. Создать приложение, показывающее картинки из пяти различных коллекций (Части света: Азия, Африка, Европа, Америка, Австралия), посвященных городам мира. Приложение должно работать так: выбираем часть света, появляется список соответствующих стран. При выборе страны из этого списка появляется его фотография флага этой страны.
16. Создать приложение, позволяющее проверить знание таблицы умножения у школьника.
17. Создать приложение, позволяющее рассчитать подоходный налог с зарплаты сотрудника.
18. Создать приложение, позволяющее посмотреть фотографию выбранного из списка сотрудника конкретного отдела.

Обязательным условием подготовки студентов к практическим занятиям является повторение ранее изученного материала по дисциплине, чтение рекомендованной дополнительной литературы. Текущий контроль усвоения знаний осуществляется путем подготовки и сдачи отчетов по итогам выполнения практических работ.

Тематика рефератов и рекомендации по их оформлению

1. Объем от 16 стр.
2. Шрифт Times New Roman.
3. Размер 14.
4. Интервал 1,5.
5. Абзацный отступ стандартный (без отступов до и после абзаца).
6. Содержание реферата: Титульный лист, оглавление (автоматическое), Введение, главы 1,2, ..., заключение, список литературы.
7. Реферат сдается в двух вариантах: напечатанном и в электронном виде.

Темы рефератов:

1. Экспертная система анализа качества исходного кода программного обеспечения
2. Современные архитектурные практики в разработке ПО
3. Архитектурные ошибки в процессе создания ПО
4. Обзор инструментальных средств разработки ПО
5. Исторический обзор развития методологии объектно-ориентированного анализа и проектирования
6. Диаграммы моделирования в языке UML
7. Диаграмма классов в языке UML
8. Диаграмма деятельности в языке UML
9. Диаграмма состояний в языке UML
10. Диаграмма вариантов использования в языке UML
11. Диаграмма последовательности в языке UML
12. История развития CASE-средств
13. Порождающие шаблоны проектирования. Назначение порождающих шаблонов. Предпосылки к использованию порождающих шаблонов проектирования.
14. Структурные шаблоны проектирования. Назначение структурных шаблонов проектирования. Структурные шаблоны уровня класса и уровня объекта.
15. Шаблоны поведения. Назначение шаблонов поведения.
16. Каскадная стратегия разработки программных средств и систем
17. Эволюционная стратегия разработки программных средств и систем
18. Проектирование графического пользовательского интерфейса
19. Современные способы построения интерфейсов
20. Методы и средства разработки пользовательского интерфейса: современное состояние.
21. Примеры разработки пользовательского графического интерфейса в сфере информационных технологий.

Контрольные вопросы для самопроверки

1. Архитектура системы – структуры классов и объектов системы. Объектно-ориентированное программирование. Эволюция объектной модели. Объектно-ориентированная декомпозиция.
2. Принципы объектной модели – абстрагирование, инкапсуляция, модульность, иерархичность, типизация.
3. Принципы реализации абстрактных типов данных – инкапсуляция, наследование и полиморфизм.
4. Класс как расширение понятия структуры. Область действия класса и доступ к членам класса.
5. Отделение интерфейса от реализации.
6. Управление доступом к членам класса
7. Конструкторы, деструкторы.
8. Перегрузка конструктора. Конструктор по умолчанию. Конструктор копирования. Конструктор преобразования. Операторы преобразования. Друзья класса.
9. Спецификаторы доступа – собственный (закрытый), общедоступный (открытый) и защищенный. Компонентные данные и компонентные функции.
10. Статические компоненты класса. Указатели на компоненты класса. Определение компонентных функций.
11. Указатель this. Дружественные компоненты класса.
12. Общие принципы перегрузки операторов. Операторные функции. Бинарные и унарные операторы. Предопределенный смысл операторов.
13. Операторы и типы, определяемые пользователем.
14. Наследование классов. Базовые и производные классы.
15. Конструкторы производных классов. Иерархии классов и объектов.
16. Одиночное наследование.
17. Множественное наследование и виртуальные базовые классы.
18. Полиморфизм времени выполнения (динамический полиморфизм).
19. Виртуальные функции. Абстрактные классы. Иерархии классов и абстрактные классы.
20. Вложенные и локальные классы.
21. Интегрированная среда.
22. Генераторы кода/приложений.
23. Шаблоны. Определение шаблонов функций.
24. Параметры шаблонов функций. Выведение типа параметров шаблона по типам аргументов при вызове функции. Переопределение шаблонов функций. Определение шаблонов классов. Параметры шаблонов классов. Создание объектов по шаблонам. Включение конструкторов в шаблон функции. Параметризация и наследование.
25. Полиморфизм времени компиляции (параметрический полиморфизм).
26. Основные концепции – контейнеры, итераторы и алгоритмы. Фундаментальные последовательности – вектора, списки, очереди с двумя концами (деки). Обзор операций с последовательностями. Адаптеры последовательностей – стеки, очереди, очереди с приоритетом. Ассоциативные контейнеры. Обзор операций с ассоциативными контейнерами. Алгоритмы и объекты-функции. Библиотеки программ и классов.

27. Обзор алгоритмов стандартной библиотеки. Итераторы и распределители памяти.

Тестовые вопросы для самоконтроля

Образцы тестов:

- 1) Стандарт пользовательского интерфейса обеспечивает...
 - a) унификацию действий приложений;
 - b) экономию времени пользователей, затрачиваемого на обучение;
 - c) сокращение времени проектирования;
 - d) унификацию приложений;
 - e) унификацию проектов.
- 2) Пользовательский интерфейс - это...
 - a) правила взаимодействия программ;
 - b) правила общения пользователя с приложением;
 - c) набор команд операционной системы;
 - d) правила общения пользователя с операционной системой;
 - e) правила общения с компьютер
- 3) Инкапсуляция - это:
 - a) основной принцип в объектно-ориентированном программировании;
 - b) способ описания атрибутов класса;
 - c) способ описания методов класса;
 - d) способ связывания атрибутов и методов для формирования объектов;
 - e) определение правил для управления доступом к членам.
- 4) Спецификатор доступа - это:
 - a) обязательное ключевое слово для атрибутов класса;
 - b) обязательное ключевое слово доступа к методам класса;
 - c) ключевое слово языка программирования, определяющее получение доступа к членам класса;
 - d) ключевое слово доступа к переменным экземпляра класса;
 - e) ключевое слово применяемое программистом по своему усмотрению.
- 5) Спецификатор доступа `private`: определяет:
 - a) атрибуты, доступные из `main`;
 - b) атрибуты и процедуры, доступные только методам, определенным в классе;
 - c) атрибуты и процедуры, доступные процедурам, определенным в классе и его наследнике;
 - d) доступ из наследуемого класса;
 - e) доступ к переменным экземпляра класса.
- 6) К какому компоненту класса доступ возможен только после его инициализации?
 - a) члену- данному со спецификатором доступа `public`::;
 - b) члену- данному со спецификатором доступа `private`::;
 - c) к переменной экземпляра класса;
 - d) члену- данному со модификатором доступа `static`;
 - e) члену- данному со спецификатором доступа `protected`::.
- 7) Полиморфизм реализуется с помощью:

- a) разнообразия атрибутов при описании абстракции;
 - b) разнообразия методов, характеризующих поведения абстракции;
 - c) перегрузки методов в иерархии наследования;
 - d) виртуальных функций;
 - e) перегрузки методов или с помощью виртуальных функций в иерархии наследования.
- 8) Каким образом для дружественной функции осуществляется доступ к закрытым элементам класса?
- a) через указатель на объект класса;
 - b) через обращение к функции- члену класса;
 - c) через объект этого класса;
 - d) через свой параметр;
 - e) через объект этого класса, который объявлен внутри функции или передан ей.
- 9) Функции можно перегружать благодаря:
- a) отличиям в числе или типе их параметров;
 - b) отличиям в их именах или типе их результатов;
 - c) отличиям в числе их параметров;
 - d) возможности наследования в иерархии классов;
 - e) особенностям реализации.
- 10) Благодаря обработке исключительных ситуаций можно:
- a) упростить управление и реакцию на ошибки во время выполнения программ;
 - b) сделать программу надежной и устойчивой к ошибкам;
 - c) обеспечить программу встроенным механизмом обработки ошибок;
 - d) структурировать текст программы;
 - e) обеспечить нормальное завершение программы.
- 11) Наиболее важное применение оператора dynamic- cast:
- a) упрощает логику программы;
 - b) упрощает логику приведения типа указателя базового класса к типу указателя производного класса;
 - c) обеспечивает более безопасный и интуитивно понятный способ выполнения операции преобразования типа;
 - d) в языках программирования, в которых реализуется полиморфизм.
- 12) Что образует ядро библиотеки стандартных шаблонов?
- a) контейнеры;
 - b) алгоритмы;
 - c) итераторы;
 - d) контейнеры, алгоритмы и итераторы;
 - e) распределители памяти, предикаты и функции сравнения.

VIII. Перечень педагогических и информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (или модулю), включая перечень программного обеспечения и информационных справочных систем (по необходимости)

Процесс изучения дисциплины включает лекции, практические занятия и самостоятельную работу студента. Во время обучения применяется контактная

технология преподавания (за исключением самостоятельно изучаемых студентами вопросов). При проведении занятий применяется имитационный подход (метод деловой игры, анализ конкретных ситуаций), когда преподавателем разбирается на конкретном примере проблемная ситуация, все шаги решения задачи студентам демонстрируются при помощи мультимедийной техники. Затем студенты самостоятельно решают аналогичные задания. Так же при проведении занятий применяется частично-поисковый метод: студенты осуществляют поиск решения поставленной проблемы (задачи). При этом постановочные задачи опираются на уже имеющиеся у студентов знания и умения, полученные в предшествующих темах. На занятиях практикуется выполнение заданий в малых группах, письменные работы, работа с раздаточным материалом, привлекаются ресурсы сети Интернет. Курс предусматривает выполнение тестов, контрольных и самостоятельных работ, письменных домашних заданий. В качестве форм контроля используются различные варианты взаимопроверки и взаимоконтроля.

Программное обеспечение:

Наименование специальных* помещений и помещений для самостоятельной работы	Перечень лицензионного программного обеспечения. Реквизиты подтверждающего документа
<p>Учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Учебная аудитория № 224 (Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Google Chrome бесплатно Kaspersky Endpoint Security 10 для Windows Акт на передачу прав ПК545 от 16.12.2022 Lazarus бесплатно OpenOffice бесплатно Многофункциональный редактор ONLYOFFICE бесплатное ПО бесплатно ОС Linux Ubuntu бесплатное ПО бесплатно</p>
<p>Помещение для самостоятельной работы, учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Компьютерный класс математического факультета № 16 (Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Google Chrome бесплатно Kaspersky Endpoint Security 10 для Windows Акт на передачу прав ПК545 от 16.12.2022 Lazarus бесплатно OpenOffice бесплатно Многофункциональный редактор ONLYOFFICE бесплатное ПО бесплатно ОС Linux Ubuntu бесплатное ПО бесплатно</p>

<p>Помещение для самостоятельной работы, учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Компьютерный класс математического факультета № 21</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Google Chrome бесплатно Kaspersky Endpoint Security 10 для Windows Акт на передачу прав ПК545 от 16.12.2022 Lazarus бесплатно OpenOffice бесплатно Многофункциональный редактор ONLYOFFICE бесплатное ПО бесплатно ОС Linux Ubuntu бесплатное ПО бесплатно</p>
<p>Учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Учебная аудитория. Математический кабинет</p> <p>№ 213</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Google Chrome бесплатно Kaspersky Endpoint Security 10 для Windows Акт на передачу прав ПК545 от 16.12.2022 Lazarus бесплатно OpenOffice бесплатно Многофункциональный редактор ONLYOFFICE бесплатное ПО бесплатно ОС Linux Ubuntu бесплатное ПО бесплатно</p>

IX. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине (или модулю)

<p>Наименование специальных* помещений и помещений для самостоятельной работы</p>	<p>Оснащенность специальных помещений и помещений для самостоятельной работы</p>
<p>Учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций,</p>	<p>Набор учебной мебели, меловая доска, Переносной ноутбук,</p> <p>Мультимедийный проектор BenQ MP 724 с потолочным</p>

<p>текущего контроля и промежуточной аттестации,</p> <p>Учебная аудитория № 224</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>креплением и экраном 1105</p>
<p>Помещение для самостоятельной работы, учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Компьютерный класс математического факультета № 16</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Набор учебной мебели, меловая доска,</p> <p>Компьютер INT Allegro, монитор Benq 24" GL2460 – 10 шт.</p>
<p>Помещение для самостоятельной работы, учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации,</p> <p>Компьютерный класс математического факультета № 21</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>Набор учебной мебели,</p> <p>Компьютер iRU Corp 510 I5-2400/4096/500/G210-512/DVD-RW/W7S/монитор E-Machines E220HQVB 21.5" – 8 шт.;</p> <p>Коммутатор D-Link DGS-1016D/GE</p>
<p>Учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной</p>	<p>Набор учебной мебели, меловая доска, Переносной ноутбук,</p> <p>Компьютер:(процессор Core i5-2400+монитор LC E2342T (10шт.)</p> <p>Графопроектор, мультимедийный комплект учебного класса (вариант № 1) Проектор Casio</p>

<p>аттестации,</p> <p>Учебная аудитория. Математический кабинет</p> <p>№ 213</p> <p>(Корпус 3, 170002, Тверская обл., г.Тверь, пер. Садовый, дом 35)</p>	<p>XJ-M140, кронштейн, кабель, удлинитель, настенный проекц. экран Lumien 180*180.</p>
--	--

Х. Сведения об обновлении рабочей программы дисциплины (или модуля)

№п. п.	Обновленный раздел рабочей программы дисциплины (или модуля)	Описание внесенных изменений	Дата и протокол заседания кафедры, утвердившего изменения
1.	V. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	Обновление списка литературы.	Протокол № 11 от 26.06.2013
2.	VII. Методические указания для обучающихся по освоению дисциплины	Корректировка планов практических (семинарских) занятий и методических рекомендаций к ним.	Протокол № 10 от 24.06.2014
3.	V. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	Обновление списка литературы. Обновление ссылок из ЭБС.	Протокол № 1 от 27.09.2015
4.	VII.	Корректировка планов	Протокол № 1 от

	Методические указания для обучающихся по освоению дисциплины.	практических (семинарских) занятий и методических рекомендаций к ним.	01.09.2016
5.	I - X	Корректировка всех разделов в соответствии с новым стандартом	Протокол № 6 от 28.02.2017
6.	V. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	Дополнение списков. Обновление ссылок из ЭБС.	Протокол № 1 от 01.09.2017
7.	V. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	Дополнение списков. Обновление ссылок из ЭБС.	Протокол № 1 от 01.09.2023