ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ МЕЖДУНАРОДНЫХ ОТНОШЕНИЙ (УНИВЕРСИТЕТ) МИНИСТЕРСТВА ИНОСТРАННЫХ ДЕЛ РОССИЙСКОЙ ФЕДЕРАЦИИ»
ТАШКЕНТСКИЙ ФИЛИАЛ

УТВЕРЖДАЮ

Директор Ташкентского филиала
МГИМО МИД России
_____ М.Т. Бакоев
«31» мая 2021 г.

Рабочая программа дисциплины

ПРОГРАММНЫЙ ПРОЕКТ

Направление подготовки
38.03.05 Бизнес-информатика

Направленность (профиль) подготовки
Анализ и моделирование социально-экономических процессов

Квалификация – б*акалавр*

Форма обучения - *очная*

Ташкент – 2021

Рабочая программа по дисциплине «Программный проект» составлена в соответствии с требованиями образовательного стандарта высшего образования МГИМО по направлению подготовки 38.03.05 Бизнес-информатика.

Авторы программы:

Вартанян Аревшад Апетович, профессор, доктор экономических наук, профессор

Каримова Венера Аркиновна, доцент, кандидат технических наук

Библиотекарь: _____ С.К. Атаханова

Содержание

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

Цель дисциплины – формирование знаний в области создания компонентов программных комплексов и баз данных, автоматизации технологических процессов с использованием современных инструментальных средств и технологий программирования.

Задачи дисциплины:

– формирования навыков применения базовых и специальных знаний в области современных информационных технологий для решения экономических задач;

– ставить и решать задачи комплексного анализа, связанные с созданием программного обеспечения и информационных систем в экономике, с использованием базовых и специальных знаний в области программирования с использованием современных аналитических методов и моделей;

– формирования навыков и умений использования современных технологий разработки программного обеспечения;

– разрабатывать новые и модернизировать уже существующие информационные технологии и системы (в экономике) в соответствии с техническим заданием.

В результате освоения образовательной программы обучающийся должен овладеть следующими результатами обучения по дисциплине (модулю):

Коды компетенции	**Содержание компетенций**	**Индикаторы достижения компетенций**	**Перечень планируемых результатов обучения по дисциплине**
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	ИУК-1.2: Разрабатывает варианты решения проблемной ситуации на основе критического анализа доступных источников информации	**Знания**: - понимать цели и задачи философского знания в контексте ее связи с другими областями духовной культуры общества; -особенности исторических этапов развития философии. **Умения**: - применять методологические принципы в изучаемых науках с учетом специфики последних; - логически и аксиологически обосновывать свои выводы и заключения в сферах научного и мировоззренческого знания. **Навыки**: - владеть навыками сбора эмпирического материала, его анализа, синтеза; обобщения и абстрагирования с целью получения необходимого для теоретической работы знания; - владеть приемами рационального логического мышления, способами получения объективного знания в соответствии с законами и правилами формальной логики.

		ИУК-1.4: Применяет виды, методы и концепции системного анализа при выработке плана действий в проблемных ситуациях	**Знания**: - виды, методы и концепции системного анализа. **Умения**: - применять виды, методы и концепции системного анализа при выработке плана действий в проблемных ситуациях. **Навыки**: - способен осуществлять поиск, критический анализ и синтез информации; - способен применять системный подход для решения поставленных задач.
УК-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	ИУК-2.3: Демонстрирует навыки эффективной организации и координации этапов реализуемого проекта с целью достижения наилучшего результата при балансировании между объемом работ и ресурсами	**Знания**: - теоретические положения управления проектами. **Умения**: - использовать методы описания информации; - проводить сравнительный анализ и выбор ИКТ для решения прикладных задач. **Навыки**: - способен эффективно организовать и координировать этапы реализуемого проекта с целью достижения наилучшего результата при балансировании между объемом работ и ресурсами.
		ИУК-2.4: Понимает виды ресурсов и ограничений для решения профессиональных задач, методы, критерии и параметры представления, описания и оценки результатов/продуктов проектной деятельности	**Знания**: - методы, критерии и параметры представления, описания и оценки результатов/продуктов проектной деятельности. **Умения**: - понимает виды ресурсов и ограничений для решения профессиональных задач, методы, критерии и параметры представления, описания и оценки результатов/продуктов проектной деятельности. **Навыки**: - способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения.
УК-9	Осознает значимость и проблемы профессиональной и социальной адаптации лиц с ограниченными возможностями	ИУК-9.1: Осознает значимость и проблемы профессиональной и социальной адаптации лиц с ограниченными возможностями	**Знания**: - технологии инклюзивного обучения. **Умения**: - осознать значимость и проблемы профессиональной и социальной адаптации лиц с ограниченными возможностями. **Навыки**: - владеет основными приемами применения законодательной базы в отношении лиц с инвалидностью и ОВЗ, готовностью действовать в нестандартных ситуациях, нести социальную и этическую ответственность за принятые решения.
		ИУК-9.3: Планирует профессиональную деятельность с лицами с ограни-	**Знания**: - теории и методы работы с лицами с ограниченными возможностями в областях про-

		ченными возможностями здоровья и инвалидами	фессиональной деятельности. **Умения**: - планировать профессиональную деятельность с лицами с ограниченными возможностями здоровья и инвалидами. **Навыки**: - способен к формированию коллективной работы с лицами с ограниченными возможностями в области профессиональной деятельности.

2. Место дисциплины в структуре образовательной программы

Дисциплина относится к вариативной части образовательного цикла (Б1.В.07).

Дисциплина изучается на 3 курсе в 6-м семестре.

Освоение курса дисциплины основано на знаниях, умениях и навыках, полученных обучающимися при изучении дисциплин: «Современные офисные технологии», «Алгоритмические языки и программирование», «Операционные системы и среды», «Структуры и алгоритмы компьютерной обработки данных».

Умения и навыки, полученные в ходе изучения дисциплины необходимы для освоения учебного материала других дисциплин профессионального цикла основной образовательной программы: «Цифровая экономика», «Информационно-аналитические и распределенные интеллектуальные системы», «Web-программирование», «Корпоративные информационные системы», а также других дисциплин, где необходимо применение инфокоммуникационных информационных технологий.

3. Объем дисциплины (модуля) в зачетных единицах с указанием количества академических часов, выделенных на контактную работу обучающихся с преподавателем (по видам занятий) и на самостоятельную работу обучающихся

Общая трудоемкость (объем) дисциплины составляет 3 зачетные единицы, 108 академических часа.

3.1. Объём дисциплины по видам учебных занятий (в часах)

Вид работы	Трудоемкость	
	Академические часы	Зачетные единицы
Общая трудоемкость	**108**	**4**
Аудиторная работа, всего: **в том числе:**	**36**	
Лекции	18	
Практические занятия/семинары:	18	
Самостоятельная работа, всего: **в том числе:**	**36**	
Самоподготовка *(самостоятельное изучение лекционного материала и материала учебников, подготовка к практическим занятиям, текущему контролю и т.д.)*	36	
Контроль (подготовка к экзамену)	**36**	
Вид промежуточной аттестации	**экзамен**	

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

4.1. Разделы дисциплины (модуля) и трудоемкость по видам учебных занятий (в академических часах)

№ п/п	Раздел/тема Дисциплины	Общая трудоёмкость *(в часах)*	Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость (в часах)			Формы текущего контроля успеваемости
			аудиторные учебные занятия		самостоятельная работа обучающихся	
		всего	лекции	семинары, практические занятия		
1.	Тема 1. Методика безопасного программирования.	8	2	2	4	Устный опрос, решение практических задач. Текущий срез №1 (контрольная работа 1)
2.	Тема 2. Объектно-ориентированный C++	16	4	4	8	
3.	Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.	8	2	2	4	Устный опрос, решение практических задач. Текущий срез №2 (контрольная работа 2)
4.	Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.	8	2	2	4	
5.	Тема 5. Среды проекта: итераторы, задача N-Body, настройка.	8	2	2	4	Устный опрос, решение практических задач. Текущий срез №3 (контрольная работа 3)
6.	Тема 6. Визуализация: OpenGL, Makefiles, Большие проекты.	8	2	2	4	
7.	Тема 7. Безопасность информационной системы.	16	4	4	8	
8.	Контроль (подготовка к экзамену)	36	0	0	0	Экзамен
	ИТОГО:	108	18	18	36	

4.2 Содержание дисциплины (модуля), структурированное по темам

Тема 1. Методика безопасного программирования.

Уязвимости программного обеспечения. Примеры эксплуатации ошибки переполнения буфера. Управление памятью в Linux. Безопасность в стеках и регистрах. Стандарты безопасного кодирования CERT. Частые ошибки программирования. Эффективные методы защиты программного кода.

Тема 2. Объектно-ориентированный C++.

Подтипы. Классы и интерфейсы. Конструкторы классов. Перегрузка операций. Наследование.

Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.

Шаблоны функций. Шаблоны классов. Паттерны проектирования программного обеспечения высокого уровня.

Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.

Структура большого проекта. Модульное тестирование. Сторонние библиотеки. Проверка кода.

Тема 5. Среды проекта: итераторы, задача N-Body, настройка.

Итераторы. Контейнеры. Алгоритмы параллельных вычислений и их программирование. Настройка программного кода и компиляторов.

Тема 6. Визуализация: OpenGL, Makefiles, Большие проекты.

Функция отображения OpenGL. Создание приложений в OpenGL. Большие проекты с 3D графикой.

Тема 7. Безопасность информационной системы.

Аварийность информационной системы без сбоев её компонентов. Методика обеспечения безопасности информационной системы. STAMP-анализ безопасности программного проектирования. Теоретико-системный подход к безопасности в программных системах.

Практические занятия и семинары.

Тема 1. Методика безопасного программирования.

1. Уязвимости программного обеспечения.
2. Примеры эксплуатации ошибки переполнения буфера.
3. Управление памятью в Linux.
4. Безопасность в стеках и регистрах.
5. Стандарты безопасного кодирования CERT.
6. Эффективные методы защиты программного кода.

Тема 2. Объектно-ориентированный C++.

1. Подтипы.
2. Классы и интерфейсы.
3. Конструкторы классов.
3. Перегрузка операций.
4. Наследование.

Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.

1. Шаблоны функций.

2. Шаблоны классов.

3. Паттерны проектирования программного обеспечения высокого уровня.

Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.

1. Структура большого проекта.
2. Модульное тестирование.
3. Сторонние библиотеки.
4. Проверка кода.

Тема 5. Среды проекта: итераторы, задача N-Body, настройка.

1. Итераторы.
2. Контейнеры.
3. Алгоритмы параллельных вычислений и их программирование.
4. Настройка программного кода и компиляторов.

Тема 6. Визуализация: OpenGL, Makefiles, Большие проекты.

1. Создание приложений в OpenGL.
2. Большие проекты с 3D графикой.

Тема 7. Безопасность информационной системы.

1. Аварийность информационной системы без сбоев её компонентов.
2. Методика обеспечения безопасности информационной системы.
3. STAMP-анализ безопасности программного проектирования.

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

Содержание самостоятельной работы	Кол-во часов	Формы самостоятельной работы
Тема 1. Методика безопасного программирования.	4	Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 2. Объектно-ориентированный C++	8	Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.	4	Проработка лекционного курса и литературы при подготовке к практическим занятиям. Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.	4	Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 5. Среды проекта: итераторы, задача N-Body, настройка.	4	Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 6. Визуализация: OpenGL, Makefiles, Боль-	4	Проработка лекционного курса и литературы при подготовке к практическим занятиям. Самостоятель-

шие проекты.		ное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.
Тема 7. Безопасность информационной системы.	8	Проработка лекционного курса и литературы при подготовке к практическим занятиям. Самостоятельное освоение теоретического материала по отдельным вопросам, чтение и проработка дополнительной литературы.

Основная часть самостоятельной работы должна включать самоподготовку студентов с использованием учебной литературы согласно списку литературы, приведенному в Рабочей программе по указанной дисциплине.

Студент должен самостоятельно освоить разделы, указанные в рабочей программе для самостоятельной работы. Как правило, эти разделы включают в себя темы дисциплины, на которые в курсе читаемых лекций уделялось недостаточное внимание, либо эти разделы не включены в курс лекций, а должны осваиваться студентом самостоятельно. В разделы самостоятельной работы студентов также включаются наиболее сложные для понимания части дисциплины, требующие более детального и углубленного изучения и осмысления.

Студент должен найти в учебной литературе соответствующую тему, прочитать ее и попытаться изложить устно или письменно основные положения или идеи прочитанного раздела.

Далее студент должен составить сам письменно вопросы, отражающие основные положения разбираемой темы и устно (или письменно) ответить на них.

Во многих рекомендуемых учебных пособиях в конце каждого раздела имеются тесты или уже сформулированные вопросы, на которые студент должен самостоятельно ответить.

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

6.1 Паспорт фонда оценочных средств по дисциплине (модулю)

1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

№ п/п	Контролируемые разделы (темы) дисциплины (результаты по разделам)	Код контролируемой компетенции и ее формулировка	Индикаторы достижения компетенции	Наименование оценочного средства
1.	Тема 1. Методика безопасного программирования.	УК-1: Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач. УК-2: Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений. УК-9: Осознает значимость и проблемы профессиональной и социальной адаптации лиц с ограниченными возможностями.	ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	Устный опрос, решение практических задач. Текущий срез №1 (контрольная работа 1)
2.	Тема 2. Объектно-ориентированный C++			
3.	Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.		ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	Устный опрос, решение практических задач. Текущий срез №2 (контрольная работа 2)
4.	Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.			
5.	Тема 5. Среды проекта: итераторы, задача N-Body, настройка.		ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	Устный опрос, решение практических задач. Текущий срез №3 (контрольная работа 3)
6.	Тема 6. Визуализация: OpenGL, Makefiles, Большие проекты.		ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	
7.	Тема 7. Безопасность информационной системы.		ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	

2а) Описание показателей и критериев оценивания компетенций на различных этапах их формирования

№ п/п	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
1.	Устный опрос	Продукт самостоятельной работы обучающегося, представляющий собой публичное выступление по представлению полученных результатов решения определённой учебно-практической, учебно-исследовательской или научной темы.	Перечень вопросов для обсуждения
2.	Решение практических задач	Проблемное задание, в котором обучающемуся предлагают осмыслить реальную профессиональ-	Практические ситуации по те-

		но-ориентированную ситуацию, необходимую для решения данной проблемы.	ме семинара
3.	Контрольная работа	Письменная работа, состоящая из нескольких вопросов.	Список вопросов для контрольной работы

2б) Описание шкал оценивания

Общий критерий оценки контрольной работы	A (90-100%)	Работа (письменный ответ) полностью отвечает целям/задачам обучения по данному курсу
	B (82-89%)	Работа (письменный ответ) в основном отвечает целям/задачам обучения по данному курсу
	C (75-81%)	Работа (письменный ответ) отвечает отдельным целям/задачам обучения по данному курсу, однако имеет серьезные недостатки в отношении остальных целей/задач
	D (67-74%)	Работа (письменный ответ) не отвечает большинству или всем целям/задачам обучения по данному курсу
	E (60-66%)	Работа (письменный ответ) совершенно не соответствует/противоречит целям данного курса; и/или не достигла их
Устный ответ	A (90-100%)	Самостоятельное и оригинальное осмысление материала; ясное и убедительное рассуждение; мощный и убедительный анализ
	B (82-89%)	Четкость логики и анализа, некоторая оригинальность в осмыслении материала, в целом работа хорошо аргументирована и убедительна
	C (75-81%)	Удовлетворительные построение и анализ при отсутствии оригинальности или критического осмысления материала
	D (67-74%)	Логика слабая, оригинальность отсутствует и/или материал недостаточно критически осмыслен
	E (60-66%)	Логика крайне слабая, отсутствует или неадекватна выбранной теме
Решение практических задач	A (90-100%)	Обучающийся решил задачу верно, без логических и арифметических ошибок, ответ обосновал и исчерпывающе аргументировал
	B (82-89%)	Обучающийся решил задачу, однако допустил некоторые арифметические ошибки, ответ обосновал
	C (75-81%)	Обучающийся решил задачу, однако допустил некоторые логические и арифметические ошибки, ответ недостаточно обоснован
	D (67-74%)	Обучающийся решил задачу неверно, допустил серьезные логические и арифметические ошибки, ответ попытался обосновать
	E (60-66%)	Обучающийся задачу не решил
Работа на занятиях	A (90-100%)	На занятиях оцениваются индивидуальные устные ответы на вопросы у доски или с места и письменные опросы. Точные, логичные ответы, быстрое и безошибочное выполнение заданий. Активен.

	B (82-89%)	Хорошо формулирует свои мысли, достаточно быстро и правильно выполняет текущие задания. Активен.
	C (75-81%)	То же, что и предыдущем пункте, только не столь безошибочно и не так быстро. Недостаточно инициативен.
	D (67-74%)	В ответах на вопросы допускает ошибки. Задания выполняет с ошибками. На занятиях неактивен.
	E (60-67%)	В ответах на вопросы допускает грубые ошибки. Несвязно излагает свои мысли. Пассивен на занятиях.
Общие умения	A (90-100%)	В ответах на вопросы допускает грубые ошибки либо отказывается от ответа. Пассивен на занятиях либо пропускает их.
	B (82-89%)	Проявлено владение достаточно широким спектром соответствующих умений
	C (75-81%)	Проявлено владение удовлетворительным спектром соответствующих умений
	D (67-74%)	Использованы отдельные общие умения, они применяются слабо или неадекватно
	E (60-67%)	Работа показывает недостаточную компетентность в области общих умений, крайне слабая работа на занятиях

3) Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков в ходе проведения промежуточной аттестации

Типовые теоретические вопросы для самоконтроля

1. Методы выявления классов.
2. Критерии проверки правильности построения класса.
3. Обработка исключительных ситуаций.
4. Типы многозадачности. Преимущества и недостатки.
5. Технология COM. Понятие интерфейса.
6. COM-сервер. Определение. Типы COM-серверов.
7. Модели STA и MTA. Преимущества и недостатки каждой модели.
8. Компонент CLR.
9. Промежуточный язык MSIL.
10. Выполнение .NET программы.
11. Понятие сборки. GAC.
12. Дать определение тестированию и отладке. Особенности и объекты тестирования. Автономное и комплексное тестирование.
13. Дать определение тестированию и отладке. Направления тестирования. Стратегия тестирования. Контрольный лист тестирования модуля.
14. Дать определение тестированию и отладке. Локализация ошибок. Классификация ошибок. Безопасное программирование.
15. Оценки ошибок.

Типовые практические задания для самоконтроля.

Тема 1. Методика безопасного программирования.
Задание 1. Исправьте следующий уязвимый код:

```
1  #include<string.h>
2
3  #define goodPass "GOODPASS"
4
5  int main() {
6    char passIsGood=0;
7    char buf[80];
8
9    printf("Enter password:\n");
10   gets(buf);
11
12   if(strcmp(buf,goodPass)==0)
13     passIsGood=1;
14   if (passIsGood == 1)
15     printf("You win!\n");
16 }
```

Решение:

```c
#include<string.h>
#include<stdio.h>

#define goodPass "GOODPASS"
#define STRSIZE 80

int main() {
  char passIsGood=0;
  char buf[STRSIZE+1];

  printf("Enter password:\n");
  fgets(buf,STRSIZE,stdin);

  if(strncmp(buf,goodPass,STRSIZE)==0)
    passIsGood=1;
  if (passIsGood == 1)
    printf("You win!\n");
}
```

Тема 2. Объектно-ориентированный C++

Задание 2. Использование библиотеки связанных списков C ++ (cpplist).

Ваша задача состоит в том, чтобы провести рефакторинг кода C и создать гораздо более гибкую библиотеку с аналогичными функциями на языке C ++. Загрузите заархивированную папку из файла cpplist.zip в качестве основы вашей программы и посмотрите на существующий код.

• Напишите свой код реализации в файлах .cpp в каталоге src /; не стесняйтесь добавлять дополнительные файлы заголовков по мере необходимости в каталог include /.

• GRADER INFO.txt - файл для грейдера (не редактируйте!), Содержащий PROG: cpplist, LANG: C ++

• Как и прежде: вы должны отправить заархивированную папку, используя ту же структуру каталогов, что и предоставленный файл. Для вашего удобства мы добавили раздел в Make-файл: если вы наберете make zip в той же папке, что и ваш проект, zip-файл, содержащий весь ваш код и необходимые заголовки, будет создан в каталоге проекта, и вы можете загрузить его к грейдеру.

Вам предоставляется файл заголовка, описывающий интерфейс для структуры данных List. Посмотрите в файле list.h, чтобы найти функциональность, необходимую для других функций, которые вы напишите.

```
// Forward declaration of apply/reduce types
class ApplyFunction;
class ReduceFunction;

class List {
  // ... put whatever private data members you need here
  // can also add any private member functions you'd like
public:
  List();
  ~List();
  size_t length() const;
  int& value( size_t pos );
  int value( size_t pos ) const;
  void append( int value );
  void deleteAll( int value );
  void insertBefore( int value, int before );
  void apply( const ApplyFunction &interface );
  int reduce( const ReduceFunction &interface ) const;
  void print() const;
};
// ..etc
```

Решение.
Файл badswitch.cpp

```
#include <cmath>
#include <iostream>

// OLD, NON-OBJECT ORIENTED
// WAY OF DOING THINGS
// DO NOT WRITE CODE LIKE THIS!

enum ShapeType {
  CIRCLE = 0,
  SQUARE = 1,
  RECTANGLE = 2,
  TRIANGLE = 3
};

struct Shape {
  ShapeType type;
  double a, b, c, d;
};

double area( const Shape &shape ) {
```

```
  switch( shape.type ) {
    case CIRCLE: return M_PI * shape.a * shape.a;
    case SQUARE: return shape.a * shape.a;
    case RECTANGLE: return shape.a * shape.b;
    case TRIANGLE: return 0.5 * shape.a * shape.b;
    default: std::cerr << "Error, invalid shape!\n";
  }
  return 0.0;
}

int main() {
  Shape circle; circle.type = CIRCLE;
  circle.a = 1.0;
  std::cout << "Circle of radius " << circle.a;
  std::cout <<  " has area " << area( circle ) << "\n";

  Shape tri; tri.type = TRIANGLE;
  tri.a = 2.0; // height
  tri.b = 3.0; // base
  std::cout << "Triangle has area ";
  std::cout << area( tri ) << "\n";
  return 0;
}
```

Файл isa.cpp

```
class Bird {
};

class FlyingBird : public Bird {
public:
  virtual void fly() = 0;
}

class Penguin : public Bird {
  // penguins are birds
  // but they shouldn't fly!
};

class AfricanSwallow : public FlyingBird {
  void fly() { /*... */ }
}
class EuropeanSwallow : public FlyingBird {
  void fly() { /*... */ }
}
```

Файл shape.cpp

```
#include <cmath>
#include <iostream>

// These class definitions would really be in another
```

```
// header file, but they're all in here for demonstration
// purposes.

class Shape {
public:
  virtual double area() const = 0;
  virtual ~Shape() {}
};

class Circle : public Shape {
  double _radius;
public:
  Circle( double theRadius ) : _radius{theRadius} {}
  ~Circle() {}
  inline double radius() const { return _radius; }

  double area() const {
    return _radius * _radius * M_PI;
  }
};

class Square : public Shape {
  double _side;
public:
  Square( double sideLength ) : _side{sideLength} {}
  ~Square() {}
  inline double side() const { return _side; }

  double area() const {
    return _side * _side;
  }
};

class Rectangle : public Shape {
  double _width, _height;
public:
  Rectangle( double theWidth, double theHeight )
    : _width{theWidth}, _height{theHeight} {}
  ~Rectangle() {}
  inline double height() const { return _height; }
  inline double width() const { return _width; }

  double area() const {
    return _width * _height;
  }
};

class Triangle : public Shape {
  double _base, _height;
public:
  Triangle( double theBase, double theHeight )
    : _base{theBase}, _height{theHeight} {}
  ~Triangle() {}
```

```
  inline double base() const { return _base; }
  inline double height() const { return _height; }

  double area() const {
    return 0.5 * _base * _height;
  }
};

int main() {
  auto circle = Circle{ 1.0 };
  auto tri = Triangle{ 2.0, 3.0 };

  std::cout << "Circle of radius ";
  std::cout << circle.radius();
  std::cout <<  " has area ";
  std::cout << circle.area() << "\n";

  std::cout << "Triangle has area ";
  std::cout << tri.area() << "\n";

  return 0;
}
```

Файл stlqueue.cpp

```
#include <queue>
#include <stack>
#include <iostream>

int main() {
  std::queue<float> floatQueue;
  std::stack<float> floatStack;

  for( int i = 0; i < 10; ++i ) {
    floatQueue.push( (float) i );
    floatStack.push( (float) i );
  }

  std::cout << "Stack FILO: ";
  while( !floatStack.empty() ) {
    std::cout << floatStack.top() << " ";
    floatStack.pop();
  }
  std::cout << "\n";

  std::cout << "Queue FIFO: ";
  while( !floatQueue.empty() ) {
    std::cout << floatQueue.front() << " ";
    floatQueue.pop();
  }
  std::cout << "\n";

  return 0;
```

```
}
```

Файл stlvector.cpp

```
#include <vector>
#include <iostream>

int main() {
  std::vector<int> intArray;

  for( int i = 0; i < 10; ++i ) {
    intArray.push_back( i * i );
  }

  // for val in list:
  // use auto &val if you want
  // to modify that position
  for( auto &val : intArray ) {
    std::cout << val << " ";
    val = 5;
  }
  std::cout << "\n";

  // use plain auto val if you want
  // to just access the value
  for( auto val : intArray ) {
    std::cout << val << " ";
  }
  std::cout << "\n";
  return 0;
}
```

Файл vector.cpp

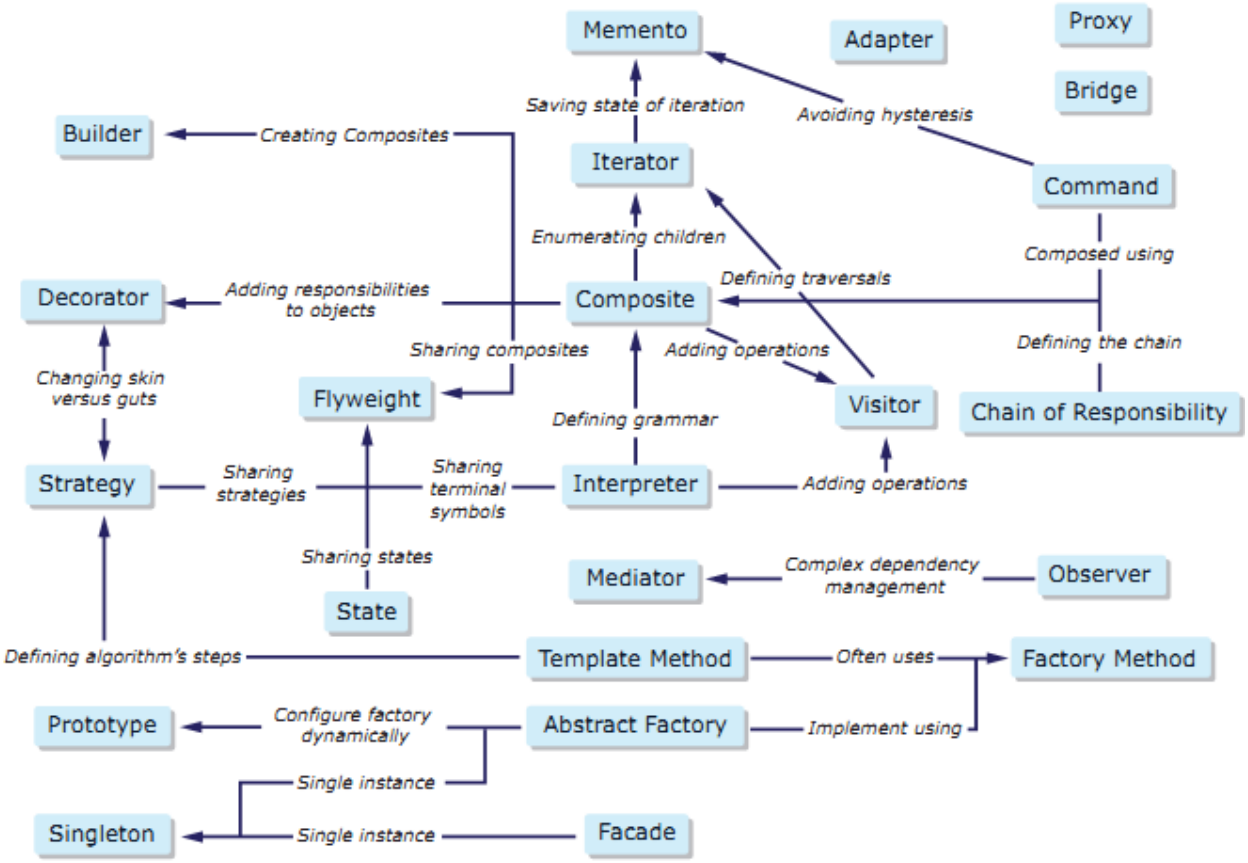
```
#include <iostream>

// -------------- Interface

class Array {
  size_t _size;
  double *_elem;
public:
  Array( size_t theSize );
  Array( const Array& );
  Array& operator=( const Array& );
  ~Array();
  Array& getSelf() {
    return *this;
  }
  bool amIthis( Array *other );

  //void resize( size_t newSize );
```

```
  inline size_t size() const { return _size; };

  double& operator[]( size_t i ) const;
  //double operator[] ( size_t i ) const;
  void print() const;
};

// -------------- Implementation
Array::Array( size_t theSize )
  :   _size{theSize},
      _elem{new double[theSize]} {
  for( size_t i = 0; i < _size; ++i ) {
    _elem[i] = 0;
  }
}

Array::Array( const Array &rhs )
  : _size{rhs._size}, _elem{nullptr} {

    _elem = new double[_size];
    for( size_t i = 0; i < _size; ++i ) {
      _elem[i] = rhs._elem[i];
    }
}
// if we're doing Array a2 = oldArray;
//                             ^rhs
// what if we write:
// oldArray = oldArray;
Array& Array::operator=( const Array &rhs ) {
  // NOT if( *this != rhs ) {
  if( this != &rhs ) {
    _size = rhs._size;
    _elem = new double[_size];
    for( size_t i = 0; i < _size; ++i ) {
      _elem[i] = rhs._elem[i];
    }
  }
  return *this;
}

bool Array::amIthis( Array *other ) {
  return this == other;
}

Array::~Array() {
  delete [] _elem;
}

double& Array::operator[]( size_t i ) const {
  return _elem[i];
}
```

```
/*double Array::operator[]( size_t i ) const {
  return _elem[i];
}*/
void Array::print() const {
  std::cout << "[ ";
  for( size_t i = 0; i < _size; ++i ) {
    std::cout << _elem[i] << " ";
  }
  std::cout << "]\n";
}

int main() {
  Array array{ 10 };
  array = array;

  std::cout << array.amIthis( &array ) << "\n";

  array.print();
//  array2.print();
  return 0;
}
```

Тема 3. Шаблоны проектирования: разработка программ более высокого уровня.

Задание 1. Дайте характеристику и опишите 23 стандартных шаблонов в C++.



Тема 4. Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода.

Задание 1. Дайте характеристику и описание для всех компонентов большого проекта на C++.

```
|-- build                          ...
|   `-- project                     |-- Makefile
|-- include                         |-- src
|   `-- project                     |   |-- gcd.cpp
|-- install                         |   `-- rational.cpp
|   |-- bin                         |-- test
|   |-- include                     |   |-- project-test.cpp
|   |-- lib                         |   `-- rationalTest.cpp
|   `-- test                        `-- third_party
|-- make                                `-- gtest
|   |-- all_head.mk
|   |-- all_tail.mk
|   |-- project.mk
|   `-- project-test.mk
```

Тема 5. Среды проекта: итераторы, задача N-Body, настройка.

Задание 1. N-Body Gravity Simulation. Имеют N точечных масс с начальными положениями ri, скоростями vi, ускорениями ai, и массы mi. Вычислить все пары сил.

Решение.

Файл Body.h

```
#ifndef _NBODY_BODY_H
#define _NBODY_BODY_H

#include <nbody/Vector3.h>

#include <iosfwd>

namespace nbody {

  class Body {
    Vector3f _position;
    Vector3f _velocity;
    Vector3f _force;
    float _mass;
  public:
    Body() : _position{}, _velocity{}, _force{}, _mass{} {}
    inline Vector3f position() const { return _position; }
    inline Vector3f& position() { return _position; }
```

```
        inline Vector3f velocity() const { return _velocity; }
        inline Vector3f& velocity() { return _velocity; }
        inline Vector3f force() const { return _force; }
        inline Vector3f& force() { return _force; }
        inline float mass() const { return _mass; }
        friend std::istream& operator>>( std::istream &is, Body
&body );
        friend std::ostream& operator<<( std::ostream &os, const
Body &body );
      };

    } // namespace nbody

    #endif // _NBODY_BODY_H
```

Файл constants.h

```
    #ifndef _NBODY_CONSTANTS_H
    #define _NBODY_CONSTANTS_H

    #include <cstddef>

    namespace nbody {

      extern const size_t MAX_BODIES_RECOMMENDED; // =
    10000
      extern const float NEWTON_G; // = 1.0f

    } // namespace nbody
    #endif // _NBODY_CONSTANTS_H
```

Файл Simulation.h

```
    #ifndef _NBODY_SIMULATION_H
    #define _NBODY_SIMULATION_H

    #include <nbody/System.h>

    #include <iosfwd>
    #include <string>

    namespace nbody {

      class Simulation {
        System *_system;
        std::string _name;
        Simulation( const Simulation& sim ) = delete;
        Simulation& operator=( const Simulation& sim ) =
    delete;
        std::string generateName();
      public:
        Simulation() : _system{nullptr}, _name{ generate-
```

```
Name() } {}
        Simulation( std::istream &input ) : _system{new
System(input)}, _name{ generateName() } {}
        void evolveSystem( int nSteps, float dt );
        void loadRun( std::istream &input );
        void saveRun() const;
      };

    } // namespace nbody

#endif // _NBODY_SIMULATION_H
```

Файл System.h

```
    #ifndef _NBODY_SYSTEM_H
    #define _NBODY_SYSTEM_H
    #include <nbody/Vector3.h>
    #include <nbody/Body.h>
    #include <iosfwd>
    #include <string>
    namespace nbody {
      class System {
        size_t _nBodies;
        Body *_body;
        float _softFactor = 1e-9f;
        float _dampingFactor = 1.0f;
        System() = delete;
        System( const System &sys ) = delete;
        System& operator=( const System &sys ) = delete;
      public:
        System( size_t N ) : _nBodies{N}, _body{ new
Body[N] } { initRandomState(); }
        System( std::istream &input ) : _nBodies{},
_body{nullptr} { readState( input ); }
        System( std::string filename ) : _nBodies{},
_body{nullptr} { readState( filename ); }
        ~System() { delete [] _body; }
        void interactBodies( size_t i, size_t j, float
softFactor, Vector3f &acc ) const;
        void computeGravitation();
        void integrateSystem( float dt );
        void readState( std::istream &input );
        void readState( std::string filename );
        void writeState( std::ostream &output ) const;
        void writeState( std::string filename ) const;
        void initRandomState();
        void update( float dt );
        void setSoftening( float soft ) { _softFactor =
soft; }
        void setDamping( float damp ) { _dampingFactor =
damp; }
      };
    } // namespace nbody
#endif // _NBODY_SYSTEM_H
```

Файл Vector3.h

```
#ifndef _VECTOR3_H
#define _VECTOR3_H
#include <cmath>
#include <iosfwd>

template<typename T>
class Vector3 {
  T _x, _y, _z;
public:
  Vector3() : _x{}, _y{}, _z{} {}
  Vector3( T x_, T y_, T z_ ) : _x{x_}, _y{y_}, _z{z_}
{}
  inline T x() const { return _x; }
  inline T y() const { return _y; }
  inline T z() const { return _z; }
  T norm() const;
  T normsq() const;
  friend std::istream& operator>>( std::istream &is,
Vector3<T> &vec ) {
    is >> vec._x >> vec._y >> vec._z;
    return is;
  }
};
template<typename T>
inline T Vector3<T>::normsq() const {
  return _x * _x + _y * _y + _z * _z;
}
template<typename T>
inline T Vector3<T>::norm() const {
  return sqrt( normsq() );
}
template<>
inline float Vector3<float>::norm() const {
  return sqrtf( normsq() );
}
template<typename T>
inline const Vector3<T> operator+( const Vector3<T> &a,
const Vector3<T> &b ) {
  return Vector3<T>{ a.x() + b.x(), a.y() + b.y(),
a.z() + b.z() };
}
template<typename T>
inline const Vector3<T> operator-( const Vector3<T> &a,
const Vector3<T> &b ) {
  return Vector3<T>{ a.x() - b.x(), a.y() - b.y(),
a.z() - b.z() };
}
// Vector * scalar
template<typename T>
inline const Vector3<T> operator*( const Vector3<T> &a,
```

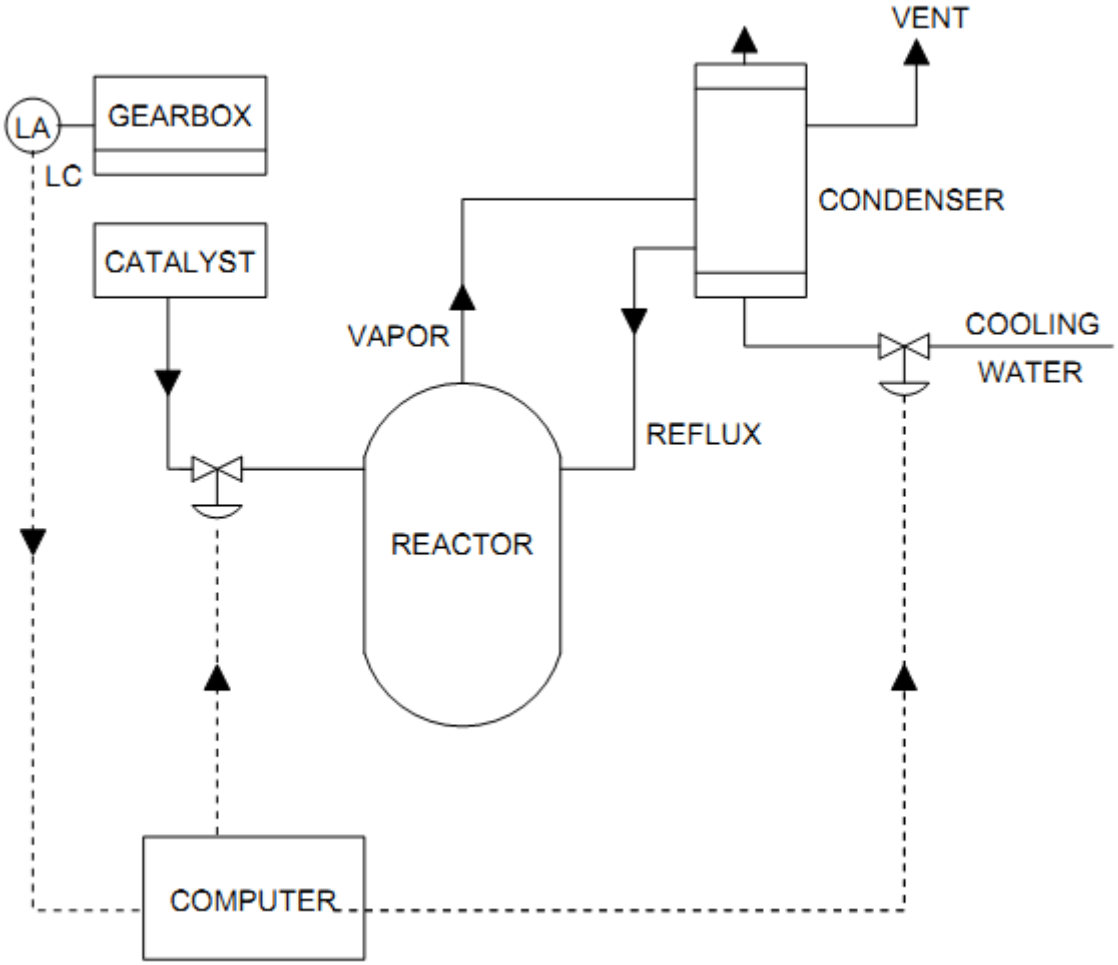
```
T b ) {
       return Vector3<T>{ a.x() * b, a.y() * b, a.z() * b };
     }

     // scalar * Vector
     template<typename T>
     inline const Vector3<T> operator*( T a, const Vec-
tor3<T> &b ) {
       return Vector3<T>{ a * b.x(), a * b.y(), a * b.z() };
     }
     // Vector / scalar
     template<typename T>
     inline const Vector3<T> operator/( const Vector3<T> &a,
T b ) {
       return Vector3<T>{ a.x() / b, a.y() / b, a.z() / b };
     }
     // Other useful math
     template<typename T>
     inline T dot( const Vector3<T> &a, const Vector3<T> &b
) {
       return a.x() * b.x() + a.y() * b.y() + a.z() * b.z();
     }
     template<typename T>
     inline Vector3<T> cross( const Vector3<T> &a, const
Vector3<T> &b ) {
       return Vector3<T>{ a.y() * b.z() - a.z() * b.y(),
                          a.z() * b.x() - a.x() * b.z(),
                          a.x() * b.y() - a.y() * b.x() };
     }
     template<typename T>
     inline T cube( T x ) {
       return x * x * x;
     }
     template<typename T>
     std::ostream& operator<<( std::ostream &os, const Vec-
tor3<T> &vec ) {
       os << vec.x() << " " << vec.y() << " " << vec.z();
       return os;
     }
     typedef Vector3<float> Vector3f;
     typedef Vector3<double> Vector3d;
     typedef Vector3<int> Vector3i;
#endif // _VECTOR3_H
```

Тема 6. Визуализация: OpenGL, Makefiles, Большие проекты.
Задание 1. Установка формата пикселя.
Решение.

```
PIXELFORMATDESCRIPTOR pfd;
// обнуляем все только что созданной структуры;
ZeroMemory(&pfd, sizeof(pfd));
// заполняем структуру
pfd.nSize = sizeof(pfd);
pfd.nVersion = 1;
// флаги показывают, что мы будем использовать
// дублирующую буферизацию OpenGL в этом окне
Pfd.dwFlags = PFD_DRAW_TO_WINDOW |
              PFD_SUPPORT_OPENGL |
              PFD_DOUBLEBUFFER;
pfd.iPixelType = PFD_TYPE_RGBA;
// полноцветный буфер цвета (8 бит на канал)
pfd.cColorBits = 24;
// запрашиваем 16 бит на пиксель для буфера глубины
pfd.cDepthBits = 16;
pfd.iLayerType = PFD_MAIN_PLANE;
int iFormat = ChoosePixelFormat(hDC, &pfd);
SetPixelFormat(hDC, iFormat, &pfd);
```

Тема 7. Безопасность информационной системы.

Задание 1. Дайте характеристики элементов обеспечения безопасности программного средства при аварии без сбоев компонентов.



Текущий контроль знаний по дисциплине
Текущий срез №1

Вопросы для подготовки к контрольному срезу:

1. Перегрузка функций в языке C++. Статический полиморфизм. Сигнатура функции. Синтаксис объявления и вызова перегруженной функции. Примеры.

2. Создание библиотечного модуля C++, содержащего набор функций (на примере). Определение собственного пространства имен, выделение заголовочного файла модуля и файла реализации функций.

3. Понятие класса в языке C++. Отношения между классом и объектом. Определение пользовательского класса, создание объекта, доступ к полям объекта и вызов методов.

4. Инкапсуляция данных внутри класса C++. Разграничение доступа к полям и методам с помощью спецификаторов public, protected, private. Дружественные функции.

5. Конструктор и деструктор класса C++. Основные задачи конструктора и деструктора, особенности их определения и использования. Конструктор по умолчанию.

6. Взаимодействие классов: композиция и агрегация. Объявление и использование агрегатных классов в программе C++. Изображение композиции и агрегации на диаграмме классов.

7. Способы инициализации объектов. Копирование объектов. Поверхностное и глубокое копирование. Конструктор копии. «Правило Трех».

8. Поля и методы объекта в оперативной памяти. Статические поля класса, их объявление и использование. Константные методы класса. Константные объекты.

9. Перегрузка операторов в C++: оператор как глобальная функция. Синтаксис объявления операторной функции. Явный и неявный вызов. Правила перегрузки операторов. Дружественные функции-операторы.

10. Перегрузка операторов в C++: оператор как метод класса. Синтаксис объявления, явный и неявный вызов операторной функции. Рекомендации по выбору способа перегрузки оператора C++.

11. Взаимодействие классов: наследование. Отношение обобщения, примеры. Механизм повторного использования кода при наследовании.

12. Синтаксис наследования в языке C++. Объявление производного класса. Доступ к элементам родительского класса из объектов класса-наследника. Множественное наследование.

13. Разграничение прав доступа при наследовании. Частное и общее наследование. Таблица прав доступа при общем (public) наследовании. Пример разграничения прав доступа.

14. Полиморфизм включения (чистый полиморфизм) в C++. Виртуальные функции. Чистые виртуальные функции. Абстрактные классы. Виртуальный деструктор.

15. Динамическая идентификация типа объекта (RTTI) в C++. Исполь-

зование оператора dynamic_cast. Использование оператора typeid из библиотеки <typeinfo>.

16. Шаблон функции в C++ (универсальный алгоритм). Синтаксис определения шаблонной функции. Параметры шаблона. Вызов функции-шаблона. Выведение типов.

17. Шаблон класса в C++ (универсальный контейнер). Синтаксис определения шаблона класса. Создание шаблонного объекта (инстанцирование шаблона).

18. Обработка исключений в программе C++. Определение исключительной ситуации, виды исключений. Синтаксис операторов try, catch, throw.

19. Этапы разработки программного обеспечения. Каскадная и итеративная модели разработки. Сложность программного обеспечения. Признаки сложной структуры. Декомпозиция как стратегия борьбы со сложностью.

20. Объектная модель программной системы. Этапы построения объектной модели. Природа объекта и класса. Структура объектов и структура классов. Методы объектно-ориентированного анализа и проектирования.

Контрольная работа №1. Темы: «Методика безопасного программирования», «Объектно-ориентированный C++».

Задание 1. Составление классов на языке C++.

Вариант 1. Составить описание класса для представления комплексных чисел с возможностью задания вещественной и мнимой частей вещественными числами. Обеспечить выполнение операций сложения, вычитания, умножения и вывода на экран.

Вариант 2. Построить класс для описания круга на плоскости. Предусмотреть методы для создания объекта, вычисления площади, перемещения на плоскости, изменения размеров, вывода на экран характеристик круга.

Вариант 3. Построить класс для описания прямоугольного треугольника на плоскости. Предусмотреть методы для создания объекта, вычисления площади, перемещения на плоскости, изменения размеров, вывода на экран характеристик треугольника.

Вариант 4. Составить описание класса прямоугольников со сторонами, параллельны-ми осям координат. Предусмотреть возможность создания, перемещения на плоскости, изменения размеров, построения наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

Вариант 5. Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, вывод на экран описания многочлена.

Вариант 6. Составить класс для описания матрицы вещественных чисел. Предусмотреть возможность создания, изменения размеров, заполнения случайными числами, сложения матриц одинакового размера, вывода на экран.

Вариант 7. Составить класс для описания мэйла. Свойства – отправитель, получатель, текст, дата отправки (день-месяц-год). Предусмотреть методы для создания письма с задаваемыми пользователем свойствами, для записи письма в файл, чтения писем из файла, а также для поиска всех писем полученных в один и тот же месяц.

Вариант 8. Составить класс для описания мэйла. Свойства – отправитель, получатель, текст, дата отправки (день-месяц-год). Предусмотреть методы для создания письма с задаваемыми пользователем свойствами, для записи письма в файл, чтения писем из файла, а также для поиска всех писем, в которых текст письма состоит меньше чем из 50 символов.

Вариант 9. Составить класс для описания мэйла. Свойства – отправитель, получатель, текст, дата отправки (день-месяц-год). Предусмотреть методы для создания письма с задаваемыми пользователем свойствами, для записи письма в файл, чтения писем из файла, а также для поиска всех писем, отправленных одному и тому же адресату.

Вариант 10. Составить класс для описания студента. Свойства – имя, пол, возраст, статический массив длины 5, состоящий из оценок за предыдущую сессию. При задании свойств использовать enum, чтобы ограничить варианты значений пола (м, ж), возраста (16 - 25), оценок (2, 3, 4, 5). Сделать метод для создания заданного пользователем количества объектов со случайными значениями свойств, вывода на экран всех студентов и отдельно тех из них, кто сдал сессию без троек.

Вариант 11. Составить класс для описания студента. Свойства – имя, пол, возраст, статический массив длины 5, состоящий из оценок за предыдущую сессию. При задании свойств использовать enum, чтобы ограничить варианты значений пола (м, ж), возраста (16 - 25), оценок (2, 3, 4, 5). Сделать метод для создания заданного пользователем количества объектов со случайными значениями свойств, вывода на экран всех студентов и отдельно тех из них, кто старше 18 лет.

Вариант 12. Составить класс для описания студента. Свойства – имя, пол, возраст, статический массив длины 5, состоящий из оценок за предыдущую сессию. При задании свойств использовать enum, чтобы ограничить варианты значений пола (м, ж), возраста (16 - 25), оценок (2, 3, 4, 5). Сделать метод для создания заданного пользователем количества объектов со случайными значениями свойств, вывода на экран всех студентов и отдельно тех из них, у кого имя короче 6 символов.

Оценка по выполнению контрольной работы производится в соответствии с таблицей.

Оценка	Оценка/ Процент	Описание критериев оценки
Отлично	A (90-100%)	Правильно сформированы программы на C++. Получены полные ответы на теоретические вопросы.

Хорошо	B (82-89%)	В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены полные ответы на теоретические вопросы.
	C (75-81%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены ответы на 75% теоретических вопросов
Удовлетворительно	D (67-74%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ есть ошибки, связанные с расчетами. Получены ответы на 70% теоретических вопросов.
	E (60-67%)	В программах C++ отсутствовали некоторые необходимые элементы. В моделях программах C++ имеются ошибки, связанные с выводом текста и рисунков. Не получены ответы на все теоретические вопросы.

Текущий срез №2

Вопросы для подготовки к контрольным срезам:

1. Понятие паттерна проектирования
2. Описание паттернов проектирования
3. Использование языка UML для представления диаграмм классов
4. Методология решения задач проектирования с помощью паттернов
5. Выбор подходящих паттернов
6. Технология использования паттерна.
7. Рефакторинг кода
8. Паттерн Абстрактная фабрика
9. Паттерн Строитель
10. Паттерн Фабричный метод
11. Паттерн Прототип
12. Паттерн Одиночка
13. Паттерн Адаптер
14. Паттерн Мост
15. Паттерн Компоновщик
16. Паттерн Декоратор
17. Паттерн Цепочка обязанностей
18. Паттерн Команда
19. Паттерн Интерпретатор
20. Паттерн Итератор
21. Паттерн Посредник
22. Паттерн Наблюдатель
23. Архитектура проектирования.
24. Модель-Контроллер-Представление (MVC)
25. Стандартная библиотека шаблонов STL. Основные компоненты STL и методы их взаимодействия. Последовательные контейнеры – вектор, дек, связный список. Ассоциативные контейнеры – множества и упорядо-

ченные ассоциативные массивы.

26. Библиотека STL: контейнеры последовательностей (vector, deque, list). Конструктор, деструктор, функции вставки/извлечения, доступ к элементам. Обход элементов – понятие об итераторе. Примеры.

27. Библиотека STL: адаптеры стека, очереди и очереди с приоритетом. Создание стека, очереди и очереди с приоритетом. Основные функции, реализуемые этими структурами данных. Примеры.

Контрольная работа №2. Тема «Шаблоны проектирования: разработка программ более высокого уровня», «Введение в проекты: модульное тестирование, сторонние библиотеки, проверка кода».

Задание 1. Реализуйте паттерн Abstract Factory для военной стратегии "Пунические войны".

Оценка по выполнению контрольной работы производится в соответствии с таблицей.

Оценка	Оценка/ Процент	Описание критериев оценки
Отлично	A (90-100%)	Правильно сформированы программы на C++. Получены полные ответы на теоретические вопросы.
Хорошо	B (82-89%)	В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены полные ответы на теоретические вопросы.
	C (75-81%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены ответы на 75% теоретических вопросов
Удовлетворительно	D (67-74%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ есть ошибки, связанные с расчетами. Получены ответы на 70% теоретических вопросов.
	E (60-67%)	В программах C++ отсутствовали некоторые необходимые элементы. В моделях программах C++ имеются ошибки, связанные с выводом текста и рисунков. Не получены ответы на все теоретические вопросы.

Текущий срез №3

Вопросы для подготовки к контрольному срезу:

1. Итераторы. Общее описание.
2. Итераторы потоков ввода-вывода
3. Контейнеры. Общее описание
4. Типы, определенные в контейнерах. Параметры конструкторов
5. Функции-члены всех контейнеров

6. Функции-члены последовательных контейнеров

7. Дополнительные функции-члены класса list

8. Функции-члены ассоциативных контейнеров

9. Вставка и удаление в последовательных контейнерах

10. Контейнеры-адаптеры и контейнеры, добавленные в стандарт C++11 и C++14.

11. Дополнение: обратные итераторы

12. Опишите архитектуру библиотек OpenGL и организацию конвейера.

13. В чем заключаются функции библиотек, подобных GLUT или GLX? Почему они формально не входят в OpenGL?

14. Назовите категории команд (функций) библиотеки OpenGL.

15. Что можно сказать о количестве и типе параметров команды glColor4ub()? glVertex3fv()?

16. Что такое функция обратного вызова и как функции обратного вызова могут быть использованы для работы с OpenGL?

17. Для чего нужна функция обновления изображения и что она делает?

18. Что такое примитив в OpenGL?

19. Что такое атрибут? Перечислите известные вам атрибуты вершин в OpenGL.

20. Для чего в OpenGL используются команды glEnable и glDisable?

21. Что такое операторные скобки и для чего они используются в OpenGL?

22. Что такое дисплейные списки? Как определить список и как вызвать его отображение?

23. Поясните организацию работы с массивами вершин и их отличие от дисплейных списков.

24. Поясните работу команды glDrawElementsQ.

25. Какие системы координат используются в OpenGL?

26. Перечислите виды матричных преобразований в OpenGL. Каким образом в OpenGL происходят преобразования объектов?

27. Что такое матричный стек?

28. Перечислите способы изменения положения наблюдателя в OpenGL.

29. Какая последовательность вызовов команд glTranslateQ, glRotateQ и glScale () соответствует команде gluLookAt(0, 0, -10, 10, 0, 0, 0, -1, 0)?

30. Какие стандартные команды для задания проекций вы знаете?

31. Приемы работы с OpenGL. Графические алгоритмы на основе OpenGL.

32. Приемы работы с OpenGL. Оптимизация программ.

33. Создание приложений с OpenGL с помощью GLUT.

34. Использование OpenGL приложений в MFC и VCL.

35. Использование OpenGL приложений в .Net.

Контрольная работа №3. Темы: «Среды проекта: итераторы, задача

N-Body, настройка», «Визуализация: OpenGL, Makefiles, Большие проекты», «Безопасность информационной системы».

Задание 1. В GLUT приложении с использованием библиотеки OpenGL разработайте программу, которая отображает случайный набор цветных прямоугольников, которые меняют свой цвет при нажатии левой кнопки мыши. С помощью нажатия правой кнопки мыши необходимо изменять режим заливки прямоуголдьников.

Оценка по выполнению контрольной работы производится в соответствии с таблицей.

Оценка	Оценка/ Процент	Описание критериев оценки
Отлично	A (90-100%)	Правильно сформированы программы на C++. Получены полные ответы на теоретические вопросы.
Хорошо	B (82-89%)	В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены полные ответы на теоретические вопросы.
	C (75-81%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ имелись незначительные ошибки, которые были устранены в ходе защиты. Получены ответы на 75% теоретических вопросов
Удовлетворительно	D (67-74%)	В программах C++ отсутствовали некоторые необходимые элементы. В программах C++ есть ошибки, связанные с расчетами. Получены ответы на 70% теоретических вопросов.
	E (60-67%)	В программах C++ отсутствовали некоторые необходимые элементы. В моделях программах C++ имеются ошибки, связанные с выводом текста и рисунков. Не получены ответы на все теоретические вопросы.

Вопросы для подготовки к экзамену:

1. Перегрузка функций в языке C++. Статический полиморфизм. Сигнатура функции. Синтаксис объявления и вызова перегруженной функции. Примеры.

2. Создание библиотечного модуля C++, содержащего набор функций (на примере). Определение собственного пространства имен, выделение заголовочного файла модуля и файла реализации функций.

3. Понятие класса в языке C++. Отношения между классом и объектом. Определение пользовательского класса, создание объекта, доступ к полям объекта и вызов методов.

4. Инкапсуляция данных внутри класса C++. Разграничение доступа к полям и методам с помощью спецификаторов public, protected, private. Дружественные функции.

5. Конструктор и деструктор класса C++. Основные задачи конструктора и деструктора, особенности их определения и использования. Конструктор по умолчанию.

6. Взаимодействие классов: композиция и агрегация. Объявление и использование агрегатных классов в программе C++. Изображение композиции и агрегации на диаграмме классов.

7. Способы инициализации объектов. Копирование объектов. Поверхностное и глубокое копирование. Конструктор копии. «Правило Трех».

8. Поля и методы объекта в оперативной памяти. Статические поля класса, их объявление и использование. Константные методы класса. Константные объекты.

9. Перегрузка операторов в C++: оператор как глобальная функция. Синтаксис объявления операторной функции. Явный и неявный вызов. Правила перегрузки операторов. Дружественные функции-операторы.

10. Перегрузка операторов в C++: оператор как метод класса. Синтаксис объявления, явный и неявный вызов операторной функции. Рекомендации по выбору способа перегрузки оператора C++.

11. Взаимодействие классов: наследование. Отношение обобщения, примеры. Механизм повторного использования кода при наследовании.

12. Синтаксис наследования в языке C++. Объявление производного класса. Доступ к элементам родительского класса из объектов класса-наследника. Множественное наследование.

13. Разграничение прав доступа при наследовании. Частное и общее наследование. Таблица прав доступа при общем (public) наследовании. Пример разграничения прав доступа.

14. Полиморфизм включения (чистый полиморфизм) в C++. Виртуальные функции. Чистые виртуальные функции. Абстрактные классы. Виртуальный деструктор.

15. Динамическая идентификация типа объекта (RTTI) в C++. Использование оператора dynamic_cast. Использование оператора typeid из библиотеки <typeinfo>.

16. Шаблон функции в C++ (универсальный алгоритм). Синтаксис определения шаблонной функции. Параметры шаблона. Вызов функции-шаблона. Выведение типов.

17. Шаблон класса в C++ (универсальный контейнер). Синтаксис определения шаблона класса. Создание шаблонного объекта (инстанцирование шаблона).

18. Обработка исключений в программе C++. Определение исключительной ситуации, виды исключений. Синтаксис операторов try, catch, throw.

19. Этапы разработки программного обеспечения. Каскадная и итеративная модели разработки. Сложность программного обеспечения. Признаки сложной структуры. Декомпозиция как стратегия борьбы со сложностью.

20. Объектная модель программной системы. Этапы построения объектной модели. Природа объекта и класса. Структура объектов и структура классов. Методы объектно-ориентированного анализа и проектирования.

21. Понятие паттерна проектирования
22. Описание паттернов проектирования
23. Использование языка UML для представления диаграмм классов
24. Методология решения задач проектирования с помощью паттернов
25. Выбор подходящих паттернов
26. Технология использования паттерна.
27. Рефакторинг кода
28. Паттерн Абстрактная фабрика
29. Паттерн Строитель
30. Паттерн Фабричный метод
31. Паттерн Прототип
32. Паттерн Одиночка
33. Паттерн Адаптер
34. Паттерн Мост
35. Паттерн Компоновщик
36. Паттерн Декоратор
37. Паттерн Цепочка обязанностей
38. Паттерн Команда
39. Паттерн Интерпретатор
40. Паттерн Итератор
41. Паттерн Посредник
42. Паттерн Наблюдатель
43. Архитектура проектирования.
44. Модель-Контроллер-Представление (MVC)
45. Стандартная библиотека шаблонов STL. Основные компоненты STL и методы их взаимодействия. Последовательные контейнеры – вектор, дек, связный список. Ассоциативные контейнеры – множества и упорядоченные ассоциативные массивы.
46. Библиотека STL: контейнеры последовательностей (vector, deque, list). Конструктор, деструктор, функции вставки/извлечения, доступ к элементам. Обход элементов – понятие об итераторе. Примеры.
47. Библиотека STL: адаптеры стека, очереди и очереди с приоритетом. Создание стека, очереди и очереди с приоритетом. Основные функции, реализуемые этими структурами данных. Примеры.
48. Итераторы. Общее описание.
49. Итераторы потоков ввода-вывода
50. Контейнеры. Общее описание
51. Типы, определенные в контейнерах. Параметры конструкторов
52. Функции-члены всех контейнеров
53. Функции-члены последовательных контейнеров
54. Дополнительные функции-члены класса list
55. Функции-члены ассоциативных контейнеров
56. Вставка и удаление в последовательных контейнерах
57. Контейнеры-адаптеры и контейнеры, добавленные в стандарт C++11 и C++14.

58. Дополнение: обратные итераторы

59. Опишите архитектуру библиотек OpenGL и организацию конвейера.

60. В чем заключаются функции библиотек, подобных GLUT или GLX? Почему они формально не входят в OpenGL?

61. Назовите категории команд (функций) библиотеки OpenGL.

62. Что можно сказать о количестве и типе параметров команды glColor4ub()? glVertex3fv()?

63. Что такое функция обратного вызова и как функции обратного вызова могут быть использованы для работы с OpenGL?

64. Для чего нужна функция обновления изображения и что она делает?

65. Что такое примитив в OpenGL?

66. Что такое атрибут? Перечислите известные вам атрибуты вершин в OpenGL.

67. Для чего в OpenGL используются команды glEnable и glDisable?

68. Что такое операторные скобки и для чего они используются в OpenGL?

69. Что такое дисплейные списки? Как определить список и как вызвать его отображение?

70. Поясните организацию работы с массивами вершин и их отличие от дисплейных списков.

71. Поясните работу команды glDrawElementsQ.

72. Какие системы координат используются в OpenGL?

73. Перечислите виды матричных преобразований в OpenGL. Каким образом в OpenGL происходят преобразования объектов?

74. Что такое матричный стек?

75. Перечислите способы изменения положения наблюдателя в OpenGL.

76. Какая последовательность вызовов команд glTranslateQ, glRotateQ и glScale () соответствует команде gluLookAt(0, 0, -10, 10, 0, 0, 0, -1, 0)?

77. Какие стандартные команды для задания проекций вы знаете?

78. Приемы работы с OpenGL. Графические алгоритмы на основе OpenGL.

79. Приемы работы с OpenGL. Оптимизация программ.

80. Создание приложений с OpenGL с помощью GLUT.

81. Использование OpenGL приложений в MFC и VCL.

82. Использование OpenGL приложений в .Net.

б) критерии оценивания компетенций (результатов)

Максимальная оценка 100%, в том числе:

- ответ на вопрос № 1 – от 0% до 50%;
- ответ на вопрос № 2 – от 0% до 50%.

в) шкала соответствия оценок

Оценка по пятибалльной шкале	Рейтинговая оценка, %	Европейская оценка
«Отлично» (5)	90-100 %	A
«Хорошо» (4)	82-89 %	B
	75-81 %	C
«Удовлетворительно» (3)	67-74 %	D
	60-66 %	E
«Неудовлетворительно» (2)	Менее 60 %	F

в) описание шкалы оценивания

Итоговым результатом считается оценка, полученная студентом по результатам работы в семестре (выставляется на основании результатов контрольных работ, тестов, защиты курсовой работы и работы на семинарских занятиях), проставленная преподавателем в зачетной ведомости. Студент может получить интегральную оценку от 70% до 100%.

4) Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Виды работы	**Индикаторы компетенций, проверяемые в процессе выполнения данного вида работы**	**Доля вида работы в итоговой оценке**
Контрольная работа №1 – промежуточная аттестация № 1	Тема 1-7. ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	от 0% до 25%
Контрольная работа №2 – промежуточная аттестация № 2		от 0% до 25%
Контрольная работа №3 – промежуточная аттестация № 3		от 0% до 25%
Работа на семинарах	Ответы на вопросы преподавателя по теме семинара, выполнение домашних заданий, основанных на лекционном материале. ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	от 0% до 25%
Итог (Экзамен)	Итоговым результатом по курсу считается оценка, полученная студентом на экзамене. Ответ студента оценивается в % с учетом шкалы соответствия рейтинговых оценок пятибалльным и европейским оценкам. ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	от 0% до 100%
Экзамен	Ответы по экзаменационным билетам. ИУК-1.2; ИУК-1.4; ИУК-2.3; ИУК-2.4; ИУК-9.1; ИУК-9.3	от 0% до 100%

7. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины (модуля)

а) Основная литература:

1. Технологии и методы программирования: учебное пособие для прикладного бакалавриата / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Юрайт, 2019. – 235 с.: (Серия: Бакалавр. Прикладной курс). – https://biblio-online.ru/bcode/433611..
2. Программирование графики на C++. Теория и примеры: учеб. пособие / В.И. Корнеев, Л.Г. Гагарина, М.В. Корнеева. – М.: ИД «ФОРУМ»: ИНФРА-М, 2017. – 517 с. – URL: https://znanium.com/catalog/product/562914.
3. Проектирование современных баз данных: Учебно-методическое пособие / Э.Г. Дадян. – М.: НИЦ ИНФРА-М, 2017. – 120 с. – URL: https://znanium.com/catalog/product/959294.
4. Проектирование информационных систем: учеб. пособие / В.В. Коваленко. – М.: ФОРУМ: ИНФРА-М, 2018. – 320 с. – URL: https://znanium.com/catalog/product/980117.
5. Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 400 с. – URL: https://znanium.com/catalog/product/1011120.

б) Дополнительная литература:

1. Программные средства и механизмы разработки информационных систем: Учебное пособие / А.А. Лежебоков. – Таганрог: Южный федеральный университет, 2016. – 86 с. – URL: https://znanium.com/catalog/product/997088.
2. Введение в архитектуру программного обеспечения: Учебное пособие / Гагарина Л.Г., Федоров А.Р., Федоров П.А. – М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2017. – 320 с. – URL: https://znanium.com/catalog/product/615207.
3. Управление качеством программного обеспечения: учебник / Б.В. Черников. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 240 с. – URL: https://znanium.com/catalog/product/1018037.
4. Язык C++ и объектно-ориентированное программирование в C++. Лабораторный практикум: Учебное пособие для вузов / Ашарина И.В., Крупская Ж.Ф. – М.: Гор. линия-Телеком, 2016. – 232 с. – URL: https://znanium.com/catalog/product/973780.
5. Скрапинг веб-сайтов с помощью Python / Р. Митчелл; пер. с англ. А. В. Груздева. – Москва : ДМК Пресс, 2016. - 280 с. – Режим доступа: URL: http://znanium.com/catalog/product/1027754.

в) Интернет-ресурс, базы данных:

1. Система федеральных образовательных порталов. Информационно-коммуникационные технологии в образовании. http://www.ict.edu.ru/lib/.

2. Интернет университет информационных технологий. http://www.intuit.ru/.

3. Система федеральных образовательных порталов. Информационно-коммуникационные технологии в образовании. http://www.ict.edu.ru/lib/.

4. Российская национальная библиотека (РНБ). www.hbl-russia.ru.

5. Российская государственная библиотека (РГБ). http://www.rsl.ru.

6. ЭБС «Университетская библиотека онлайн» http:www.biblioclub.ru.

7. ЭБС «Znanium.com» http://znanium.com.

8. ЭБС «Юрайт» https://biblio-online.ru.

9. CIT-Forum. Обзор паттернов проектирования - http://citforum.ru/SE/project/pattern/

10. Source Making (eng) - http://sourcemaking.com/

11. Паттерны проектирования. Раздел на сайте для программистов Cpp-Reference - http://cpp-reference.ru/patterns/

12. Справочник. Паттерны проектирования. - http://design-pattern.ru

8. Методические указания для обучающихся по освоению дисциплины (модуля)

Вид учебных занятий	Организация деятельности студента
Лекция	Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения; помечать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, пометить и попытаться найти ответ в рекомендуемой литературе. Если самостоятельно не удается разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, на практическом занятии. Уделить внимание следующим понятиям (*перечисление понятий*) и др.
Практические занятия	Проработка рабочей программы, уделяя особое внимание целям и задачам структуре и содержанию дисциплины. Конспектирование источников. Работа с конспектом лекций, подготовка ответов к контрольным вопросам, просмотр рекомендуемой литературы, работа с текстом (*указать текст из источника и др.)*. Прослушивание аудио- и видеозаписей по заданной теме, решение расчетно-графических заданий, решение задач по алгоритму и др.
Контрольная работа / индивидуальные задания	Знакомство с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующих для запоминания и являющихся основополагающими в этой теме. Составление аннотаций к прочитанным литературным источникам и др.
Подготовка к экзамену	При подготовке к экзамену необходимо ориентироваться на конспекты лекций, рекомендуемую литературу и др.

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

Перечень лицензионного программного обеспечения:

1) операционная система Windows;

2) пакет Microsoft Office;

3) антивирус Nod32;

4) справочная правовая система КонсультантПлюс

5) антивирус Nod32; архиватор 7z., Ramus Educational, Aris Express, Bizagi Process Modeller

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Учебная аудитория для проведения занятий всех видов, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации.
Технические средства обучения:
Мультимедиа-проектор-1шт.
LCD экран для демонстрации презентаций – 1 шт
Монитор преподавателя- 1 шт.
Системный блок-1шт.
Комплект аудио колонок для воспроизведения аудио файла-1шт.
Специализированная мебель:
Доска-1шт
Стол преподавателя-1шт.
Стол студенческий одноместный-25 шт.
Стулья студенческие -25 шт.
Компьютеры марки Avtech – 25 шт.

11. Иные сведения и материалы

12. Лист регистрации внесенных изменений