МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)

УТВЕРЖДАЮ
Директор департамента образования

Документ подписан электронной подписью
Сертификат: 1c6cfa0a-52a6-4f49-aef0-5584d3fd4820
Владелец: Троян Павел Ефимович
Действителен: с 19.01.2016 по 16.09.2019

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Технологии программирования

Уровень образования: **высшее образование - бакалавриат**
Направление подготовки / специальность: **09.03.04 Программная инженерия**
Направленность (профиль) / специализация: **Проектирование и разработка программных продуктов**
Форма обучения: **заочная**
Факультет: **ЗиВФ, Заочный и вечерний факультет**
Кафедра: **АОИ, Кафедра автоматизации обработки информации**
Курс: **4**
Семестр: **7, 8**
Учебный план набора 2014 года

Распределение рабочего времени

№	Виды учебной деятельности	7 семестр	8 семестр	Всего	Единицы
1	Лекции	4	4	8	часов
2	Лабораторные работы	0	16	16	часов
3	Всего аудиторных занятий	4	20	24	часов
4	Самостоятельная работа	104	187	291	часов
5	Всего (без экзамена)	108	207	315	часов
6	Подготовка и сдача экзамена	0	9	9	часов
7	Общая трудоемкость	108	216	324	часов
				9.0	З.Е.

Контрольные работы: 8 семестр - 1
Экзамен: 8 семестр

Томск 2018

Рассмотрена и одобрена на заседании кафедры
протокол № 318 от « 3 » 7 2018 г.

ЛИСТ СОГЛАСОВАНИЯ

Рабочая программа дисциплины составлена с учетом требований федерального государственного образовательного стандарта высшего образования (ФГОС ВО) по направлению подготовки (специальности) 09.03.04 Программная инженерия, утвержденного 12.03.2015 года, рассмотрена и одобрена на заседании кафедры АОИ «___» _____ 20__ года, протокол №_____.

Разработчик:

Старший преподаватель каф. АОИ _____ И. В. Безходарнов

Заведующий обеспечивающей каф. АОИ _____ Ю. П. Ехлаков

Рабочая программа дисциплины согласована с факультетом и выпускающей кафедрой:

Декан ЗиВФ _____ И. В. Осипов

Заведующий выпускающей каф. АОИ _____ Ю. П. Ехлаков

Эксперты:

Доцент кафедры автоматизации обработки информации (АОИ) _____ Н. Ю. Салмина

Доцент кафедры автоматизации обработки информации (АОИ) _____ А. А. Сидоров

1. Цели и задачи дисциплины

1.1. Цели дисциплины

Расширить кругозор технических знаний студентов о различных видах ПО, методах из проектирования, создания и эксплуатации

Научить студентов пользоваться различными методами проектирования и создания разных видов ПО

Познакомить студентов с технологиями проектирования, создания и эксплуатации. Дать практические навыки их использования

1.2. Задачи дисциплины

- Познакомить студентов с различными направлениями существующими в отрасли ПО (Windows, Linux, WEB, Сетевое и системное программирование, создание пользовательских интерфейсов, автоматизированное тестирование ПО, защита ПО от взлома)
- Изучить и дать навыки использования нескольких методологий создания ПО (процедурное программирование, ООП, шаблоны проектирования, графическое проектирование)
- Изучить и дать навыки использования технологий, использующихся на всем протяжении жизненного цикла ПО (методы оценки и экспертизы проекта, общее представление о жизненном цикле ПО, методы и инструменты автоматизации процессов жизненного цикла ПО)
- Дать общее представление о процессах жизненного цикла ПО

2. Место дисциплины в структуре ОПОП

Дисциплина «Технологии программирования» (Б1.В.ДВ.5.1) относится к блоку 1 (вариативная часть).

Предшествующими дисциплинами, формирующими начальные знания, являются: Информатика и программирование, Операционные системы и сети, Проектирование и архитектура программных систем.

Последующими дисциплинами являются: Управление жизненным циклом программных систем.

3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций:

- ОПК-3 готовностью применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;
- ПК-3 владением навыками использования различных технологий разработки программного обеспечения;

В результате изучения дисциплины обучающийся должен:

- **знать** Различные методы проектирования, создания и эксплуатации программного обеспечения
- **уметь** Использовать на практике различные методы и технологии проектирования и создания разные виды программного обеспечения
- **владеть** Навыками создания ПО для Windows, Linux, WEB Навыками проектирования и создания пользовательских интерфейсов Навыками применения автоматизированного тестирования Владеть навыками создания ПО в процедурной парадигме программирования и парадигме ООП Методами оценки и экспертизы проектов Методами и инструментами автоматизации процессов жизненного цикла ПО

4. Объем дисциплины и виды учебной работы

Общая трудоемкость дисциплины составляет 9.0 зачетных единицы и представлена в таблице 4.1.

Таблица 4.1 – Трудоемкость дисциплины

Виды учебной деятельности	Всего часов	Семестры	
		7 семестр	8 семестр
Аудиторные занятия (всего)	24	4	20

Лекции	8	4	4
Лабораторные работы	16	0	16
Самостоятельная работа (всего)	291	104	187
Оформление отчетов по лабораторным работам	28	12	16
Подготовка к лабораторным работам	16	0	16
Проработка лекционного материала	22	8	14
Самостоятельное изучение тем (вопросов) теоретической части курса	185	84	101
Выполнение контрольных работ	40	0	40
Всего (без экзамена)	315	108	207
Подготовка и сдача экзамена	9	0	9
Общая трудоемкость, ч	324	108	216
Зачетные Единицы	9.0		

5. Содержание дисциплины

5.1. Разделы дисциплины и виды занятий

Разделы дисциплины и виды занятий приведены в таблице 5.1.

Таблица 5.1 – Разделы дисциплины и виды занятий

Названия разделов дисциплины	Лек., ч	Лаб. раб., ч	Сам. раб., ч	Всего часов (без экзамена)	Формируемые компетенции
7 семестр					
1 Методы программирования: процедурная парадигма и парадигма ООП	4	0	104	108	ОПК-3, ПК-3
Итого за семестр	4	0	104	108	
8 семестр					
2 Технологии создания ПО	4	16	187	207	ОПК-3, ПК-3
Итого за семестр	4	16	187	207	
Итого	8	16	291	315	

5.2. Содержание разделов дисциплины (по лекциям)

Содержание разделов дисциплин (по лекциям) приведено в таблице 5.2.

Таблица 5.2 – Содержание разделов дисциплин (по лекциям)

Названия разделов	Содержание разделов дисциплины (по лекциям)	Трудоемкость, ч	Формируемые компетенции
7 семестр			
1 Методы программирования: процедурная парадигма	Вводная лекция о методах программирования. Анализ Бизнес-идеи проекта по созданию программного обеспечения	2	ОПК-3

и парадигма ООП	Парадигмы программирования, практические вопросы их применения	2	
	Итого	4	
Итого за семестр		4	
8 семестр			
2 Технологии создания ПО	Основные механизмы языков программирования и объектно-ориентированного программирования	2	ОПК-3, ПК-3
	Обзорная лекция: основные технологии ПО, их описание, характеристика процессов необходимых для создания ПО в этих технологиях, описание знаний, необходимых для работы в каждом из направлений	2	
	Итого	4	
Итого за семестр		4	
Итого		8	

5.3. Разделы дисциплины и междисциплинарные связи с обеспечивающими (предыдущими) и обеспечиваемыми (последующими) дисциплинами

Разделы дисциплины и междисциплинарные связи с обеспечивающими (предыдущими) и обеспечиваемыми (последующими) дисциплинами представлены в таблице 5.3.

Таблица 5.3 – Разделы дисциплины и междисциплинарные связи

Наименование дисциплин	№ разделов данной дисциплины, для которых необходимо изучение обеспечивающих и обеспечиваемых дисциплин	
	1	2
Предшествующие дисциплины		
1 Информатика и программирование	+	
2 Операционные системы и сети		+
3 Проектирование и архитектура программных систем	+	+
Последующие дисциплины		
1 Управление жизненным циклом программных систем		+

5.4. Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий

Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий представлено в таблице 5.4.

Таблица 5.4 – Соответствие компетенций, формируемых при изучении дисциплины, и видов занятий

Компетенции	Виды занятий			Формы контроля
	Лек.	Лаб. раб.	Сам. раб.	
ОПК-3	+	+	+	Контрольная работа, Экзамен, Отчет по лабораторной работе, Тест

ПК-3	+	+	+	Контрольная работа, Экзамен, Отчет по лабораторной работе, Тест

6. Интерактивные методы и формы организации обучения

Не предусмотрено РУП.

7. Лабораторные работы

Наименование лабораторных работ приведено в таблице 7.1.

Таблица 7.1 – Наименование лабораторных работ

Названия разделов	Наименование лабораторных работ	Трудоемкость, ч	Формируемые компетенции
8 семестр			
2 Технологии создания ПО	Анализ проекта/идеи – изучение методики оценки	4	ОПК-3
	Прототипирование интерфейсов	4	
	Небольшое GUI-приложение	8	
	Итого	16	
Итого за семестр		16	
Итого		16	

8. Практические занятия (семинары)

Не предусмотрено РУП.

9. Самостоятельная работа

Виды самостоятельной работы, трудоемкость и формируемые компетенции представлены в таблице 9.1.

Таблица 9.1 – Виды самостоятельной работы, трудоемкость и формируемые компетенции

Названия разделов	Виды самостоятельной работы	Трудоемкость, ч	Формируемые компетенции	Формы контроля
7 семестр				
1 Методы программирования: процедурная парадигма и парадигма ООП	Самостоятельное изучение тем (вопросов) теоретической части курса	84	ОПК-3, ПК-3	Отчет по лабораторной работе, Тест, Экзамен
	Проработка лекционного материала	8		
	Оформление отчетов по лабораторным работам	12		
	Итого	104		
Итого за семестр		104		
8 семестр				
2 Технологии создания ПО	Выполнение контрольных работ	40	ОПК-3, ПК-3	Контрольная работа, Отчет по лабораторной ра-

	Самостоятельное изучение тем (вопросов) теоретической части курса	101		боте, Тест, Экзамен
	Проработка лекционного материала	14		
	Подготовка к лабораторным работам	16		
	Оформление отчетов по лабораторным работам	16		
	Итого	187		
Итого за семестр		187		
	Подготовка и сдача экзамена	9		Экзамен
Итого		300		

10. Курсовой проект / курсовая работа

Не предусмотрено РУП.

11. Рейтинговая система для оценки успеваемости обучающихся

Рейтинговая система не используется.

12. Учебно-методическое и информационное обеспечение дисциплины

12.1. Основная литература

1. Кьоу, Джим. Объектно-ориентированное программирование : Учебный курс. - СПб. : Питер , 2005. - 237[3] с. (наличие в библиотеке ТУСУР - 20 экз.)

2. Липаев, Владимир Васильевич. Проектирование программных средств : Учебное пособие для вузов. - М. : Высшая школа , 1990. - 301[3] с. (наличие в библиотеке ТУСУР - 12 экз.)

3. Технология разработки программного обеспечения [Электронный ресурс]: Учеб. пос. / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул; Под ред. проф. Л.Г.Гагариной - М. ИД ФОРУМ НИЦ Инфра-М, 2013. - 400 с. ил.; 60x90 1/16. - (Высшее обр.). (п) ISBN 978-5-8199-0342-1 - Режим доступа: http://znanium.com/catalog/product/389963 (дата обращения: 06.08.2018).

12.2. Дополнительная литература

1. Технология программирования на C++. Win32 API-приложения [Электронный ресурс]: Учебное пособие / Литвиненко Н.А. - СПбБХВ-Петербург, 2010. - 280 с. ISBN 978-5-9775-0600-7 - Режим доступа: http://znanium.com/catalog/product/351463 (дата обращения: 06.08.2018).

2. Розенберг, Д. Применение объектного моделирования с использованием UML и анализ прецедентов [Электронный ресурс] / Д. Розенберг, К. Скотт; Пер. с англ. - М. [Электронный ресурс]: ДМК Пресс, 2007. - 160 с. ил. - (Серия «Объектно-ориентированные технологии в программировании»). - ISBN 5-94074-050-2 - Режим доступа: http://znanium.com/catalog/product/407658 (дата обращения: 06.08.2018).

3. Дейл, Н. Программирование на C++ [Электронный ресурс] / Н. Дейл, Ч. Уимз, М. Хедингтон; Пер. с англ. - М. [Электронный ресурс]: ДМК Пресс, 2007. - 672 с. ил. - (Серия «Учебник»). - ISBN 5-93700-008-0. - Режим доступа: http://znanium.com/catalog/product/407353 (дата обращения: 06.08.2018).

12.3. Учебно-методические пособия

12.3.1. Обязательные учебно-методические пособия

1. Методы и технологии программирования [Электронный ресурс]: Методические указания к лабораторным работам и организации самостоятельной работы / И. В. Безходарнов - 2018. 22 с. - Режим доступа: https://edu.tusur.ru/publications/8517 (дата обращения: 06.08.2018).

12.3.2. Учебно-методические пособия для лиц с ограниченными возможностями здоровья и инвалидов

Учебно-методические материалы для самостоятельной и аудиторной работы обучающихся из числа лиц с ограниченными возможностями здоровья и инвалидов предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации.

Для лиц с нарушениями зрения:

- в форме электронного документа;
- в печатной форме увеличенным шрифтом.

Для лиц с нарушениями слуха:

- в форме электронного документа;
- в печатной форме.

Для лиц с нарушениями опорно-двигательного аппарата:

- в форме электронного документа;
- в печатной форме.

12.4. Профессиональные базы данных и информационные справочные системы

1. Образовательный портал университета (http://edu.tusur.ru).
2. Дополнительно к профессиональным базам данных рекомендуется использовать информационные, справочные и нормативные базы данных https://lib.tusur.ru/ru/resursy/bazy-dannyh.

13. Материально-техническое обеспечение дисциплины и требуемое программное обеспечение

13.1. Общие требования к материально-техническому и программному обеспечению дисциплины

13.1.1. Материально-техническое и программное обеспечение для лекционных занятий

Для проведения занятий лекционного типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации используется учебная аудитория с количеством посадочных мест не менее 22-24, оборудованная доской и стандартной учебной мебелью. Имеются демонстрационное оборудование и учебно-наглядные пособия, обеспечивающие тематические иллюстрации по лекционным разделам дисциплины.

13.1.2. Материально-техническое и программное обеспечение для лабораторных работ

Лаборатория «Информатика и программирование»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 428 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core 2 Duo E6550 2.3 ГГц, ОЗУ – 2 Гб, жесткий диск – 250 Гб (14 шт.);
- Меловая доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- Git 2.11.03, GNU GPLv2
- Google Chrome
- IntelliJ Community, Apache 2.0 license
- Microsoft Visual Studio 2015
- Microsoft Windows 7 Pro

Лаборатория «Операционные системы и СУБД»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения кур-

совых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 430 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core 2 Duo E6550 2.3 ГГц, ОЗУ – 2 Гб, жесткий диск – 250 Гб (12 шт.);
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- Git 2.11.03, GNU GPLv2
- Google Chrome
- IntelliJ Community, Apache 2.0 license
- Java SE Development Kit
- Microsoft Visual Studio 2015
- Microsoft Windows 7 Pro
- Mozilla Firefox

Лаборатория «Распределенные вычислительные системы»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 432а ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i5-3330 3.0 ГГц, ОЗУ – 4 Гб, жесткий диск – 500 Гб (12 шт.);
- Меловая доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- Git 2.11.03, GNU GPLv2
- Google Chrome
- IntelliJ Community, Apache 2.0 license
- Microsoft Visual Studio 2015
- Microsoft Windows 10 Pro
- Mozilla Firefox

Лаборатория «Муниципальная информатика»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 432б ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i5-2320 3.0 ГГц, ОЗУ – 4 Гб, жесткий диск – 500 Гб (12 шт.);
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

- Git 2.11.03, GNU GPLv2
- Google Chrome
- IntelliJ Community, Apache 2.0 license
- Microsoft Visual Studio 2015
- Microsoft Windows 10 Pro

– Mozilla Firefox

Лаборатория «Программная инженерия»

учебная аудитория для проведения занятий практического типа, учебная аудитория для проведения занятий лабораторного типа, помещение для курсового проектирования (выполнения курсовых работ), помещение для самостоятельной работы

634034, Томская область, г. Томск, Вершинина улица, д. 74, 409 ауд.

Описание имеющегося оборудования:

- Персональный компьютер Intel Core i3-6300 3.2 ГГц, ОЗУ – 8 Гб, жесткий диск – 500 Гб (10 шт.);
- Проектор Optoma Ex632.DLP;
- Экран для проектора Lumian Mas+Er;
- Магнитно-маркерная доска;
- Комплект специализированной учебной мебели;
- Рабочее место преподавателя.

Программное обеспечение:

– Git 2.11.03, GNU GPLv2
– Google Chrome
– IntelliJ Community, Apache 2.0 license
– Microsoft Visual Studio 2015
– Microsoft Windows 10
– Mozilla Firefox

13.1.3. Материально-техническое и программное обеспечение для самостоятельной работы

Для самостоятельной работы используются учебные аудитории (компьютерные классы), расположенные по адресам:

- 634050, Томская область, г. Томск, Ленина проспект, д. 40, 233 ауд.;
- 634045, Томская область, г. Томск, ул. Красноармейская, д. 146, 201 ауд.;
- 634034, Томская область, г. Томск, Вершинина улица, д. 47, 126 ауд.;
- 634034, Томская область, г. Томск, Вершинина улица, д. 74, 207 ауд.

Состав оборудования:

- учебная мебель;
- компьютеры класса не ниже ПЭВМ INTEL Celeron D336 2.8ГГц. - 5 шт.;
- компьютеры подключены к сети «Интернет» и обеспечивают доступ в электронную информационно-образовательную среду университета.

Перечень программного обеспечения:

- Microsoft Windows;
- OpenOffice;
- Kaspersky Endpoint Security 10 для Windows;
- 7-Zip;
- Google Chrome.

13.2. Материально-техническое обеспечение дисциплины для лиц с ограниченными возможностями здоровья и инвалидов

Освоение дисциплины лицами с ограниченными возможностями здоровья и инвалидами осуществляется с использованием средств обучения общего и специального назначения.

При занятиях с обучающимися **с нарушениями слуха** предусмотрено использование звукоусиливающей аппаратуры, мультимедийных средств и других технических средств приема/передачи учебной информации в доступных формах, мобильной системы преподавания для обучающихся с инвалидностью, портативной индукционной системы. Учебная аудитория, в которой занимаются обучающиеся с нарушением слуха, оборудована компьютерной техникой, аудиотехникой, видео-

техникой, электронной доской, мультимедийной системой.

При занятиях с обучающимися **с нарушениями зрениями** предусмотрено использование в лекционных и учебных аудиториях возможности просмотра удаленных объектов (например, текста на доске или слайда на экране) при помощи видеоувеличителей для комфортного просмотра.

При занятиях с обучающимися **с нарушениями опорно-двигательного аппарата** используются альтернативные устройства ввода информации и другие технические средства приема/передачи учебной информации в доступных формах, мобильной системы обучения для людей с инвалидностью.

14. Оценочные материалы и методические рекомендации по организации изучения дисциплины

14.1. Содержание оценочных материалов и методические рекомендации

Для оценки степени сформированности и уровня освоения закрепленных за дисциплиной компетенций используются оценочные материалы в составе:

14.1.1. Тестовые задания

ОПК-3: готовность применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов

1. Какое из указанных определений соответствует императивной парадигме программирования?

1) Программа определяет процесс вычислений посредством описания логики самого вычисления, а не управляющей логики программы

2) Программа описывает процесс вычислений как вычисление значений функций в их математическом понимании

3) Программа описывает процесс вычислений посредством описания управляющей логики программы, т.е. в виде последовательности отдельных команд, которые должен выполнить компьютер. Каждая команда является инструкцией, изменяющей состояние программы

4) Программа описывает некоторые логические правила и известные факты, которые в ходе вычислений преобразуются в другие факты, нужные, для получения результата вычислений

2. Какой из указанных языков относится к декларативной парадигме программирования?

1) C++

2) Haskell

3) Python

4) Java

3. Что такое «Переменная» в большинстве языков программирования?

1) Символьная строка в тексте программы, которая зачем-то понадобилась компилятору, и мы должны ее использовать чтобы программа заработала.

2) По сути, это символьное представление ссылки на некоторую область памяти ЭВМ, где и хранится значение (данные) переменной. Человеком оно используется для придания осмысленности и читаемости тексту программы, а компилятором как уникальный идентификатор, однозначно определяющий, где брать данные относящиеся к этому идентификатору.

3) Это конкретная ячейка памяти ЭВМ, находящаяся в области данных программы.

4) Это сущность, которая используется для разделения кода программы и данных программы.

4. Какой набор правил относится к статической переменной определенной внутри тела функции (языки C/C++/Java и подобные им)?

1) Такая переменная хранится в области данных программы и может быть использована в любом месте программы, а не только внутри тела этой функции. Таким образом, переменная мо-

жет менять свое значение после выхода из функции, где она определена.

2) Эта переменная инициализируется указанным в коде значением каждый раз при входе в функцию, где она определена, но хранится в общей области данных программы за счет чего достигается более высокая скорость ее инициализации чем нестатической переменной.

3) Эта переменная будет видна (доступна для использования) только внутри тела функции, в которой она определена и при этом для ее использования при входе в тело функции будет выделен отдельный регистр процессора, значение которого будет проинициализировано тем, что указано в операторе инициализации переменной.

4) Память под эту переменную выделяется только один раз (это может происходить на этапе компиляции/сборки или на этапе запуска). Если код программы содержит инициализацию переменной, то он выполняется только один раз при первом вызове нашей функции. Переменная сохраняет свое значение после выхода из функции. Переменная может быть использована (видна для использования) только внутри тела функции.

5. При написании программы (языки C, C++, Java и подобные им), содержащей много вложенных циклов (например, при реализации математического алгоритма), мы решили заранее определить все переменные индексов как глобальные (псевдокод):

Int i1, i2, i3, i4, i5, n, k;

Unsigned int i6, i7, i8, i9, i10, m p;

Long i20, I 21, I 22, l, q, r;

И далее используем их в программе во всех процедурах и функциях. Какое из приведенных ниже утверждений неверно?

1) Наша программа теоретически будет работать быстрее, чем если бы мы определяли нужные индексы локально внутри каждой функции, так как внутри функций не будет кода, выполняющего выделение памяти под эти переменные

2) Наша программа теоретически будет занимать на диске слегка больше места, чем если бы мы определяли эти индексы локально внутри функций, так как память под глобальные переменные будет выделена уже на этапе компиляции программы, и они будут занимать место на диске

3) Такое определение переменных сделает наш код более читаемым, а программу в целом более понятной

4) Такое определение затруднит написание программы, ее отладку, ухудшит читаемость кода и понимание программы в целом. А если мы начнем использовать эти переменные как механизм передачи данных между разными функциями, то сделаем нашу программу практически непригодной к отладке и использованию другими разработчиками

6. Что делает оператор if … else?

1) В зависимости от истинности проверяемого выражения выполняет разные блоки кода

2) В зависимости от условия возвращает истина/ложь

3) Выполняет переход по указанному адресу

4) Служит для организации циклов

7. В каких конструкциях и для чего используется оператор break (языки подобные C)?

1) В операторе if … else: для инвертированной реакции на условие

2) В операторах switch: для прекращения анализа входного значения и для выхода из тела цикла

3) Для продолжения работы программы после прерывания

4) Для организации исключения

8. Для чего используется оператор continue (языки подобные C)?

1) Продолжения работы программы

2) Запуска новой итерации цикла
3) Обработки исключительных ситуаций
4) Ожидания ввода информации от пользователя

9. Что такое реентерабельная процедура?
1) Процедура, всегда выполняющаяся без ошибок
2) Процедура, одна копия которой может выполняться несколько раз, в том числе одновременно выдавая корректный результат для каждого своего запуска
3) Процедура, использующая только глобальные для всей программы переменные
4) Процедура внешняя по отношению к нашей программе и выполняющаяся в отдельном потоке

10. Что такое перегрузка процедур (функций)?
1) Так называются процедуры и функции, которые перегружают компьютер
2) Вызов процедуры из самой себя
3) Определение нескольких процедур с одним именем, но разыми параметрами, что позволяет писать один код для алгоритмически схожих ситуаций, но имеющих отличия либо в типе, либо в количестве необходимых переменных (параметров). Компилятор на этапе сопоставления типов и количества выходных параметров сам выбирает, какую из процедур (функций) нужно вызвать, что в итоге делает код программы проще для понимания, отладки и модификации
4) Так говорят о процедуре или функции, которая перегружена параметрами без явной на то необходимости

11. Какой оператор явно используется для выхода (возврата) из функции (языки подобные C)?
1) Goto
2) Break
3) Return
4) Static

12. В каком из указанных случаев несколько переменных может иметь одно и тоже имя (C, C++, Java), но при этом возможен доступ к любой из этих переменных в любом участке программы?
1) Если они находятся в одной функции, но в разных блоках кода
2) Если они определены как Auto
3) Это невозможно ни в каком случае
4) Если они находятся в разных Namespace

13. Какие типы данных может содержать в себе структура данных (C, C++, Java и подобные им языки)?
1) Любые типы данных, но только те, размер которых заранее известен компилятору
2) Только скалярные типы данных
3) Только типы данных определенные пользователем
4) Любые типы, включая типы данных с переменной длинной (не известной на этапе компиляции)

14. Для чего предназначен механизм Namespace?
1) Для определения дополнительных заголовков файлов исходного кода
2) Для отделения разных частей программы с целью исключения пересечений имен типов,

переменных и функций между ними

3) Для разделения программы на отдельно компилируемые модули

4) Для того, чтобы компилятор мог записать информацию о Namespace в исполняемый файл для ее дальнейшего использования отладчиком

15. Какое из утверждений неверно для глобальных переменных?

1) Память под глобальные переменные выделяется на этапе компиляции/сборки программы

2) Глобальные переменные существуют все время, пока выполняется программа

3) Глобальные переменные не меняют свои значения в процессе работы программы

4) Глобальные переменные доступны из любого участка программы

16. Где существует и доступна для использования локальная переменная (для языков C, C++, Java)?

1) Внутри файла, в котором она определена

2) Внутри функции, где она определена независимо от блока кода

3) На всем протяжении работы программы

4) Только внутри блока кода, в котором она определена

17. Какое из утверждений является неверным для статических переменных?

1) Статические переменные сохраняют свои значения между вызовами процедуры, где они определены

2) Статические переменные каждый раз инициализируются при входе в процедуру, где они определены

3) Статические переменные после инициализации существуют до конца выполнения программы

4) Статические переменные видны в том блоке программы, где они определены (внутри тела функции, класса, процедуры)

18. Какое из утверждений верно для динамической переменной (языки C, C++, Java и подобные им)?

1) Память под динамическую переменную выделяется компилятором на этапе сборки программы

2) Память, выделенная под динамическую переменную, будет автоматически освобождена после выхода из блока кода, в котором определена эта переменная

3) Динамическая переменная не может быть глобальной

4) Выделением и освобождением памяти под динамическую переменную управляет программист в коде программы

19. Что означает термин «Дымовое тестирование»?

1) Поиск таких условий эксплуатации, при которых работа программы становиться невозможна

2) Начальное тестирование программного обеспечения для проверки его пригодности к дальнейшим видам тестирования

3) Тестирование в условиях задымления в помещении

4) Тестирование программного обеспечения в условиях повышенной нагрузки на электронно-вычислительную машину, на которой происходит тестирование

20. Какую оптимизацию кода программного обеспечения следует делать "по умолчанию" (в

первую очередь)?

1) С точки зрения занимаемой машинной памяти

2) С точки зрения количества строк кода

3) С точки зрения скорости работы программы

4) С точки зрения читаемости кода и простоты понимания его структуры

ПК-3: владение навыками использования различных технологий разработки программного обеспечения

1. Что означает и для чего используется Инкапсуляция в объектно-ориентированном программировании?

1) Класс как сущность исходного кода включает в себя все необходимое для своего функционирования, при этом детали функционирования (реализации) скрыты от кода внешнего, по отношению к самому классу. Это используется для упрощения жизни тем, кто будет использовать данный класс (т.е. читаемость кода, простота сопровождения, изменения в случае расширения функциональности)

2) Это термин означающий, что нужно все функции включить в один класс в программе и таким образом избежать множественности классов. Это используется для запутывания исходного кода, чтобы потенциальные взломщики не могли разобраться, как работает программа

3) Программа должна содержать в себе глобальный класс Application, который должен включать в себя все другие классы, переменные и т.д. Это используется, чтобы операционная система могла запустить вашу программу и обеспечивать ее работу

4) Одни объекты могут содержать (инкапсулировать) в себе другие объекты. Это используется, чтобы не было слишком много глобальных объектов в программе

2. Что означает Наследование и для чего оно используется в объектно-ориентированном программировании?

1) Это механизм, который позволяет иметь несколько функций с одним именем, но с разными типами аргументов и их количеством. Используется для улучшения читаемости кода т.к. не забота о вызове конкретного варианта функции лежит на компиляторе, а человек видит единый для любых типов аргументов код.

2) Наследование нужно для того, чтобы можно было объявлять одни объекты потомками других и на уровне компилятора вызывать соответствующие обработчики одних объектов при изменении других.

3) Наследование дает возможность разрабатывать одни программы на основе других. Например, если взять текстовый редактор, то наследуясь от него можно проще, чем с нуля разработать графический редактор.

4) Это механизм определения одних (производных) классов на основании других (базовых), при этом производные классы (потомки) наследуют большинство переменных и методов базового класса (предка). Это используется для переиспользования кода, уже имеющегося в классах предках, и для написания единого кода, который использует классы потомки (т.е. для реализации принципа подстановки, когда любой код будет правильно использовать класс потомок от того с которым он был написан, не подозревая об этом).

3. Что такое Полиморфизм в объектно-ориентированном программировании и для чего он используется?

1) Это термин, определяющий программы, которые в процессе выполнения меняют свое содержимое. Например, так действуют некоторые компьютерные вирусы, чтобы избежать обнаружения себя по известной сигнатуре (определенной байтовой подстроке)

2) С помощью этого механизма можно создавать объекты одних классов на основании других классов, что позволяет делать их взаимозависимыми, но при этом полиморфными

3) Это возможность написать код, использующий определенный (в том числе и абстрактный класс), а потом подсунуть этому коду совсем другой класс (и возможно не один) и данный код (за счет обработки, которую встроит компилятор) будет правильно взаимодействовать с нашим классом. Таким образом, получается, что на этапе выполнения одни классы, представляются совсем другими. Данный механизм реализуется за счет других: наследование, множественное наследование, реализация интерфейсов

4) Полиморфизм – это механизм, реализуемый компилятором, который позволяет писать один и тот же код для разных скалярных типов данных. Например, можно без явного приведения типов сложить два числа разных типов (целое и с плавающей точкой) и получить число третьего типа (например, длинное целое)

4. Что означает понятие Абстракции в объектно-ориентированном программировании (ООП), и для чего оно используется?

1) Абстракция в ООП означает, что все классы являются некоторыми абстракциями по отношению к внешнему миру или деталям технической реализации. А введение в программу лишних классов, не отражающих напрямую некоторые реальные сущности, нарушает принцип абстракции и делает программу неоптимальной с точки зрения ее архитектуры.

2) Абстракция – это класс (или даже любой другой тип данных), который представляет собой модель некоей сущности предметной области или сущности технической реализации. Он моделирует только те функции и свойства, которые необходимы для реализации этой конкретной программы (т.е. все они используются в программе и их достаточно для реализации всей бизнес-логики)

3) Абстракция достигается тем, что в качестве имен объектов (а также переменных и функций) используются соответствующие предметной области или деталям технической реализации слова и аббревиатуры. Например, индекс для цикла лучше называть “ind” а не “I”. Это используется для того, чтобы программа, хотя и являющаяся абстрактной сущностью, была просто читаемой человеком сведущим в данной предметной области.

4) Абстракция - это понятие относящиеся к компиляции программы, и оно означает, что в процессе компиляции все переменные и функции теряют свои изначальные имена (кроме тех, что специально были объявлены экспортируемыми) тем самым становясь абстрактными.

5. Какое из утверждений относительно Класса в объектно-ориентированном программировании является неверным?

1) Класс – это определяемый пользователем тип данных

2) Класс содержит в себе все необходимые данные и функции для реализации своей функциональности

3) Класс должен закрыть от внешнего мира детали своей реализации, но при этом быть открытым для расширения своей функциональности

4) Класс при выполнении программы всегда хранится в памяти в единственном экземпляре

6. Какие из указанных ниже понятий не являются элементами класса?

1) Методы

2) Интерфейсы

3) Классы потомки

4) Переменные класса

7. Какие члены класса будут являться внутренними, т.е. недоступными для вызова не изнутри кода самого класса (C++, Java и подобные им языки)?

1) Помеченные как private
2) Помеченные как protected
3) Помеченные как public
4) Помеченные как static

8. Где будут доступны для использования члены класса, помеченные как protected?
1) Только внутри самого класса
2) Только из внешнего кода
3) Внутри самого класса и классов потомков
4) Только внутри классов потомков

9. Для чего нужны исключения (exception)?
1) Для обработки ошибок, возникающих в ходе выполнения программы
2) Для исключения участков кода из программы
3) Для обработки аппаратных прерываний
4) Чтобы отслеживать несанкционированную активность других (по отношению к вашей) программ

10. Что такое HTML?
1) Текстовый формат, который позволяет определять документы, организовывать их оформление, структуру, ссылки между отдельными частями документа и другими документами и т.д.
2) Текстовый формат, который в виде дерева задает стили оформление отдельных частей документа
3) Язык программирования, который исполняется на стороне браузера
4) Формат записи текстов в виде, удобном для автоматизированной обработки

11. Чему соответствует определение: Язык программирования, программа на котором исполняется на стороне браузера?
1) CSS
2) JS
3) XML
4) XSS

12. Какое утверждение относительно структуры HTML документа:
<HTML>
<HEAD>
Элементы заголовка
</HEAD>
<BODY>
Тело документа
</BODY>
</HTML>
является верным?
1) Невозможно ответить на этот вопрос, так как в приведенном примере отсутствует содержимое раздела BODY
2) Такая структура неверна, так как в ней не определены обязательные элементы заголовка
3) Этот документ не имеет отношения к HTML
4) Это допустимая структура HTML документа

13. Какого HTML тега не существует?
1) <a …>
2) <b …>
3) <title>
4) <input …>

14. Какие из указанных утилит, не являются Unix shell?
1) sh
2) bash
3) joe
4) zch

15. Для чего предназначена unix утилита du?
1) Чтения файлов с диска
2) Архивации файлов
3) Поиска файлов определенного размера
4) Определение занятого пространства на диске

16. Для чего используется реестр Windows, где он хранится?
1) Для записи протокола действий операционной системы, хранится в текстовых файлах
2) Для хранения настроек системы, хранится в нескольких бинарных файлах
3) Это место, где сохраняется резервная копия системы, хранится на отдельном диске
4) В реестре сохраняется последние состояние системы для восстановления после перехода в режим сна (Hibernate), он хранится в корне диска C:\

17. Какая утилита используется для просмотра информации и управления сетевыми интерфейсами в Windows?
1) Nslookup
2) Ipconfig
3) flushdns
4) telnet

18. Для чего не предназначена система DNS в сети интернет?
1) Определения IP адреса хоста по имени сервера
2) Предоставления информации о почтовых серверах домена
3) Преобразования IP адреса в имя хоста
4) Обнаружения ВЕБ серверов в сети

19. Какую из нижеперечисленных функций неспособна выполнять ни одна система контроля версий?
1) Контроль за правильностью исходного кода
2) Хранение разных версий исходного кода
3) Организация работы нескольких разработчиков
4) Хранение резервных копий исходного кода

20. Является ли git децентрализованной системой контроля версий?

1) Нет, не является

2) Да, является

3) Этот термин не имеет отношения к системам контроля версий

4) Это зависит от варианта установки git на компьютер пользователя

14.1.2. Экзаменационные вопросы

1. Что включает в себя понятие «Технология программирования» (ТП), что является результатом применения ТП?

2. Что включает в себя понятие «Методология программирования»? Назовите основные методологии.

3. Опишите императивную парадигму программирования, какие методологии/языки программирования относятся к ней?

4. Опишите декларативную парадигму программирования, какие методологии/языки программирования относятся к ней?

5. Назовите основные механизмы, применяемые в процедурном программировании. Дайте их краткое описание.

6. Расскажите о правилах ограничения видимости переменных и процедур в языках программирования. Для чего используется ограничения видимости?

7. Какие переменные бывают с точки зрения времени их жизни и доступности для разных участков кода программы? Коротко опишите эти типы.

8. Что такое структуры данных и определяемые типы? Приведите пример использования.

9. Назовите основные принципы ООП, дайте их краткое описание.

10. Дайте определение Класса и Объекта в ООП, опишите их основные составляющие и их характеристики

11. Расскажите об модификаторах видимости членов класса. Опишите основные из них.

12. Что такое Исключения (exception), зачем они используются, опишите механизм действия.

13. Перечислите и кратко опишите базовые технологии WEB-Frontend программирования.

14. Опишите общую структуру HTML документа, основные HTML теги (5-10 штук).

15. Расскажите структуру ОС Unix, методы взаимодействия ее компонентов, методы межпроцессного взаимодействия в этой ОС

16. Приведите команды Unix shell которые вы знаете, команды скриптов bash.

17. Перечислите основные сценарии работы с системами контроля версий исходного кода, приведите последовательности команд на примере git для реализации этих сценариев

18. Опишите работу протокола TCP/IP, перечислите и опишите свойства протокола, важные при разработке программного обеспечения с его использованием

19. Опишите работу протокола UDP, перечислите и опишите свойства протокола, важные при разработке программного обеспечения с его использованием

20. Дайте краткую характеристику протоколов IP, SNMP, TCP, UDP их использование, взаимодействие, известные Вам особенности, перечислите и коротко опишите известные Вам прикладные протоколы

21. Опишите инфраструктуру технических средств, применяемую производства программного обеспечения, и ее отдельные элементы

14.1.3. Темы контрольных работ

Шаблоны проектирования

14.1.4. Темы лабораторных работ

Анализ проекта/идеи – изучение методики оценки

Прототипирование интерфейсов

Небольшое GUI-приложение

14.1.5. Методические рекомендации

Темы для самостоятельного изучения:

Основы объектно-ориентированного программирования и проектирования.

Шаблоны проектирования.

Чистая архитектура и принципы SOLID.
Технологии разработки ПО.

14.2. Требования к оценочным материалам для лиц с ограниченными возможностями здоровья и инвалидов

Для лиц с ограниченными возможностями здоровья и инвалидов предусмотрены дополнительные оценочные материалы, перечень которых указан в таблице 14.

Таблица 14 – Дополнительные материалы оценивания для лиц с ограниченными возможностями здоровья и инвалидов

Категории обучающихся	Виды дополнительных оценочных материалов	Формы контроля и оценки результатов обучения
С нарушениями слуха	Тесты, письменные самостоятельные работы, вопросы к зачету, контрольные работы	Преимущественно письменная проверка
С нарушениями зрения	Собеседование по вопросам к зачету, опрос по терминам	Преимущественно устная проверка (индивидуально)
С нарушениями опорно-двигательного аппарата	Решение дистанционных тестов, контрольные работы, письменные самостоятельные работы, вопросы к зачету	Преимущественно дистанционными методами
С ограничениями по общемедицинским показаниям	Тесты, письменные самостоятельные работы, вопросы к зачету, контрольные работы, устные ответы	Преимущественно проверка методами исходя из состояния обучающегося на момент проверки

14.3. Методические рекомендации по оценочным материалам для лиц с ограниченными возможностями здоровья и инвалидов

Для лиц с ограниченными возможностями здоровья и инвалидов предусматривается доступная форма предоставления заданий оценочных средств, а именно:

- в печатной форме;
- в печатной форме с увеличенным шрифтом;
- в форме электронного документа;
- методом чтения ассистентом задания вслух;
- предоставление задания с использованием сурдоперевода.

Лицам с ограниченными возможностями здоровья и инвалидам увеличивается время на подготовку ответов на контрольные вопросы. Для таких обучающихся предусматривается доступная форма предоставления ответов на задания, а именно:

- письменно на бумаге;
- набор ответов на компьютере;
- набор ответов с использованием услуг ассистента;
- представление ответов устно.

Процедура оценивания результатов обучения лиц с ограниченными возможностями здоровья и инвалидов по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

- в форме электронного документа;
- в печатной форме увеличенным шрифтом.

Для лиц с нарушениями слуха:

- в форме электронного документа;
- в печатной форме.

Для лиц с нарушениями опорно-двигательного аппарата:

- в форме электронного документа;
- в печатной форме.

При необходимости для лиц с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения может проводиться в несколько этапов.