МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (ОЦЕНОЧНЫЕ СРЕДСТВА)

Шифр	Наименование дисциплины (модуля)
Б1.О	**Технологии программирования**

Код направления подготовки	09.03.02
Направление подготовки	Информационные системы и технологии
Наименование (я) ОПОП (направленность / профиль)	Информационные технологии в образовании
Уровень образования	бакалавр
Форма обучения	очная

Разработчики:

Должность	Учёная степень, звание	Подпись	ФИО
Старший преподаватель			Боровская Елена Владимировна

Рабочая программа рассмотрена и одобрена (обновлена) на заседании кафедры (структурного подразделения)

Кафедра	Заведующий кафедрой	Номер протокола	Дата протокола	Подпись
Кафедра информатики, информационных технологий и методики обучения информатике	Рузаков Андрей Александрович	10	13.06.2019	
Кафедра информатики, информационных технологий и методики обучения информатике	Рузаков Андрей Александрович	1	10.09.2020	

Раздел 1. Компетенции обучающегося, формируемые в результате освоения образовательной программы с указанием этапов их формирования

Таблица 1 - Перечень компетенций, с указанием образовательных результатов в процессе освоения дисциплины (в соответствии с РПД)

Формируемые компетенции			
Индикаторы ее достижения	**Планируемые образовательные результаты по дисциплине**		
	знать	**уметь**	**владеть**
ОПК-6 способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий			
ОПК.6.1 Знать методы алгоритмизации, языки и технологии программирования, пригодные для практического применения в области информационных систем и технологий	З.2 Знать методы алгоритмизации, языки и технологии программирования, пригодные для практического применения в области информационных систем и технологий		
ОПК.6.2 Уметь применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач в области информационных систем и технологий		У.2 Уметь применять методы алгоритмизации, языки и технологии программирования при разработке ПО	
ОПК.6.3 Иметь навыки программирования, отладки и тестирования прототипов программно-технических комплексов задач			В.2 Иметь навыки программирования, отладки и тестирования ПО
ОПК-3 способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности			
ОПК.3.1 Знать принципы, методы и средства решения стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	З.1 принципы, базовые концепции технологий программирования		
ОПК.3.2 Уметь решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.		У.1 использовать технологии программирования при разработке систем автоматизации проектирования прикладных программ;	

ОПК.3.3 Иметь навыки подготовки обзоров, аннотаций, составления рефератов, научных докладов, публикаций и библиографии по научно-исследовательской работе с учетом требований информационной безопасности.			В.1 навыками разработки проектной документации на ПО

Компетенции связаны с дисциплинами и практиками через матрицу компетенций согласно таблице 2.

Таблица 2 - Компетенции, формируемые в результате обучения

Код и наименование компетенции	
Составляющая учебного плана (дисциплины, практики, участвующие в формировании компетенции)	**Вес дисциплины в формировании компетенции (100 / количество дисциплин, практик)**
ОПК-6 способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	
Исследование операций и методы оптимизации	20,00
Технологии программирования	**20,00**
учебная практика (ознакомительная)	20,00
производственная практика (технологическая (проектно-технологическая))	20,00
Алгоритмы и структуры данных	20,00
ОПК-3 способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	
Администрирование информационных систем	12,50
Инфокоммуникационные системы и сети	12,50
Технологии программирования	**12,50**
Управление данными	12,50
Методы и средства проектирования информационных систем и технологий	12,50
учебная практика (ознакомительная)	12,50
производственная практика (технологическая (проектно-технологическая))	12,50
Управление ИТ-проектами	12,50

Таблица 3 - Этапы формирования компетенций в процессе освоения ОПОП

Код компетенции	**Этап базовой подготовки**	**Этап расширения и углубления подготовки**	**Этап профессионально-практической подготовки**
ОПК-6	**Исследование операций и методы оптимизации, Технологии программирования, учебная практика (ознакомительная), производственная практика (технологическая (проектно-технологическая)), Алгоритмы и структуры данных**		учебная практика (ознакомительная), производственная практика (технологическая (проектно-технологическая))

ОПК-3	**Администрирование информационных систем, Инфокоммуникационные системы и сети, Технологии программирования, Управление данными, Методы и средства проектирования информационных систем и технологий, учебная практика (ознакомительная), производственная практика (технологическая (проектно-технологическая)), Управление ИТ-проектами**		учебная практика (ознакомительная), производственная практика (технологическая (проектно-технологическая))

Раздел 2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Таблица 4 - Показатели оценивания компетенций на различных этапах их формирования в процессе освоения учебной дисциплины (в соответствии с РПД)

№	Раздел	
	Формируемые компетенции	
	Показатели сформированности (в терминах «знать», «уметь», «владеть»)	**Виды оценочных средств**
1	Основные понятия технологий программирования.	
	ОПК-3 ОПК-6	
	Знать принципы, базовые концепции технологий программирования Знать знать методы алгоритмизации, языки и технологии программирования, пригодные для практического применения в области информационных систем и технологий	Тест
	Уметь использовать технологии программирования при разработке систем автоматизации проектирования прикладных программ; Уметь уметь применять методы алгоритмизации, языки и технологии программирования при разработке ПО	Тест
	Владеть навыками разработки проектной документации на ПО Владеть иметь навыки программирования, отладки и тестирования ПО	Задача
2	Реализация принципов ООП в C#. Визуальное программирование.	
	ОПК-3 ОПК-6	
	Знать принципы, базовые концепции технологий программирования Знать знать методы алгоритмизации, языки и технологии программирования, пригодные для практического применения в области информационных систем и технологий	Тест
	Уметь использовать технологии программирования при разработке систем автоматизации проектирования прикладных программ; Уметь уметь применять методы алгоритмизации, языки и технологии программирования при разработке ПО	Задача
	Владеть навыками разработки проектной документации на ПО Владеть иметь навыки программирования, отладки и тестирования ПО	Задача

Таблица 5 - Описание уровней и критериев оценивания компетенций, описание шкал оценивания

Код	**Содержание компетенции**			
Уровни освоения компетенции	**Содержательное описание уровня**	**Основные признаки выделения уровня (критерии оценки сформированности)**	**Пятибалльная шкала (академическая оценка)**	**% освоения (рейтинговая оценка)**
ОПК-6	ОПК-6 способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий			
ОПК-3	ОПК-3 способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информац...			

Раздел 3. Типовые контрольные задания и (или) иные материалы, необходимые для оценки планируемых результатов обучения по дисциплине (модулю)

1. Оценочные средства для текущего контроля

Раздел: Основные понятия технологий программирования.

Задания для оценки знаний

1. Тест:

#2/
В качестве базового элемента объектно-ориентированного программирования используются
$
Объекты
Алгоритмы
Потоки
Подпрограммы
#2/
Неверным высказыванием является
$
класс является экземпляром объекта
объекты в процессе выполнения программы создаются и уничтожаются
классы организованы иерархически
термины - «экземпляр класса» и «объект» - взаимозаменяемы
#2/
Объект не обладает
$
модульностью
состоянием
поведением
идентичностью
#2/
Перечнем всех свойств (атрибутов) данного объекта и текущим значением каждого из этих свойств характеризуется
$
состояние
модульность
поведение
идентичность
#2/
Неверным высказыванием является
$
Объект может порождать любое количество классов
Класс - это некое множество объектов, имеющих общую структуру и общее поведение
Классы и объекты - это тесно связанные понятия
Классы статичны, т.е. все их особенности и содержание определены в процессе компиляции программы
#2/
Событие это - что-то, что может изменить
$
состояние объекта
модульность объекта
поведение объекта
идентичность объекта
#2/
Операция создания и/или инициализации объекта -
$
конструктор
модификатор
селектор
итератор
деструктор
#2/
Операция, освобождающая состояние объекта и/или разрушающая сам объект
$

деструктор
модификатор
селектор
итератор
конструктор
#2/
Окончанием существования любого объекта является момент его
$
уничтожения
описания
инициализации
создания
использования
#2/
Свойство классов решать схожие по смыслу проблемы различными способами называется
$
Полиморфизмом
Абстрагированием
Идентичностью
Инкапсуляцией
Иерархией
Модульностью
#2/
Инкапсулированные в класса данные называются
$
Полями
Методами
Свойствами
#2/
Программа действий над объектом или его свойствами называется
$
Методами
Полями
Свойствами
#2/
Выберите ошибочную запись доступа к свойствам и методам объектов:
$
имя свойства . имя объекта
имя объекта . имя свойства
имя объекта . имя метода
#2/
Окно формы
$
проект Windows- окна будущей программы
предназначено для отображения свойств и событий объекта
осуществляет основные функции управления проектом создаваемой программы
это палитра компонентов
предназначено для быстрого доступа к наиболее важным опциям главного меню
предназначено для создания и редактирования текста программы

#2/
Укажите ложное высказывание:
$
Процедуры нельзя использовать повторно. Можно создавать библиотеки однократно используемых процедур.
Процедурное программирование как методика разработки программного обеспечения разделяет программу на данные и процедуры, обрабатывающие эти данные.
Процедурное программирование имеет последовательную природу. В процессе выполнения процедурной программы последовательно вызываются различные процедуры. После выполнения последней из них программа завершает свою работу.
Процедурное программирование задает общую структуру программы: 1) данные 2) и процедуры.
Использование процедур помогает разрабатывать программу, предназначенную для решения конкретной задачи. Вместо написания одного большого блока обработки данных в ходе разработки программа все время разбивается на процедуры и более мелкие подпрограммы.
#2/

Укажите ложное высказывание:
$
Как процедурное, так и модульное программирование не поддерживают многократное использование.
Модульное программирование жестко связывает данные и процедуры, обрабатывающие эти данные, в компоненты программы, называемые модулями.
Модули скрывают представление данных и внутренние методы их обработки. Однако большинство модульных языков программирования дают возможность использовать модули в процедурной среде.
Модульное программирование скрывает детали реализации алгоритмов, защищая, таким образом, данные от несовместимого или некорректного изменения.
Использование модулей также задает высокоуровневую структуру программы. Такой подход дает возможность разрабатывать структуру программы на концептуальном, поведенческом уровне, а не в терминах данных и процедур.
#2/
Укажите ложное высказывание:
$
Объектно-ориентированное программирование является развитием модульного программирования, и как следствие, так же не поддерживает многократное использование.
Объектно-ориентированное программирование — метод разработки программного обеспечения, позволяющий при моделировании программы использовать названия объектов реального мира. Оно разбивает программу на набор взаимодействующих объектов.
Объектно-ориентированное программирование является развитием модульного программирования, так как поддерживает инкапсуляцию.
По сравнению с модульным программированием, в ООП введена поддержка таких свойств объектов, как наследование (для расширения возможности многократного использования) и полиморфизм (для повышения гибкости в работе с типами данных).
#2/
Укажите свойства, которыми обладают написанные с помощью объектно-ориентированного программирования программы:
$
Естественность, Надежность, Возможность многократного использования, Удобство в сопровождении (легкость исправления ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Естественность, Надежность, Возможность однократного использования, Удобство в сопровождении (легкость исправления ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Естественность, Надежность, Возможность многократного использования, Проблемы в сопровождении (тяжело исправлять ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Научность, Надежность, Возможность многократного использования, Проблемы в сопровождении (тяжело исправлять ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
#2/
Объектно-ориентированное программирование естественно. Вместо того чтобы описывать задачу в терминах данных и процедур, при использовании объектно-ориентированного программирования можно пользоваться словарем исходной задачи. Такой подход, давая возможность использовать хорошо понятные термины, позволяет сосредоточить внимание
$
На решении задачи в терминах объектов и классов, а не на деталях реализации.
На деталях реализации, а не на решении задачи в терминах объектов и классов .
На определении процедур для обработки данных предметной области
На определении структуры модулей предметной области
#2/
Ключевое слово языка программирования, которое говорит компьютеру, какая часть программы может получить доступ к атрибутам и методам, являющимся членами класса это -
$
Спецификатор доступа
Слово NEW
Слово Constructor
Имя метода
#2/
Спецификатор доступа _____определяет атрибуты и методы, которые доступны при использовании экземпляра любого класса.
$
Public

Private
Protected
New
#2/
Спецификатор доступа _____ определяет атрибуты и методы, которые доступны только методам, определенным в данном классе.
$
Private
Public
Protected
New
#2/
Спецификатор доступа_____ определяет атрибуты и процедуры, которые могут быть наследованы другими классами (производными), но не видимы всеми классами.
$
Protected
Public
Private
New
#2/
Производный класс наследует -_____и _____члены базового класса.
$
Public и protected
Private и public
Private и protected
New и protected
#2/
На рисунке изображена иерархия классов
Какой из классов является абстрактным?

$
Shape
Oval
Hexagon
Circle
ShapesApp

#2/
На рисунке изображена иерархия классов
Какой из классов является производным от абстрактного класса?

$
Circle
Shape
Oval
ShapesApp
#2/
На рисунке изображена диаграмма классов. Укажите класс являющийся интерфейсом

$
IGameObject и IPlayer
Application
Smith
#2/
На рисунке изображена диаграмма классов. Укажите класс, использующий интерфейс

$
Smith
IGameObject и IPlayer
Application
#2/
Каждый язык программирования определяется
$
Алфавитом, синтаксисом, семантикой

Символами, словами и предложениями
Правилами организации и структурирования
#4/
Расставьте методологии программирования в соответствии с этапами развития технологий программирования
$
Методологии программирования нет, программирование считается искусством.
Методология - структурный подход.
Методология - модульный подход.
Методология - объектно-ориентированный подход.
#2/
Под семантикой языка программирования понимают
$
правила, определяющие какие операции, и в какой последовательности должна выполнять ЭВМ, работающая по программе, т.е. это смысловое значение слов и фраз написанных на языке программирования.

набор символов, включающий буквы, цифры и специальные знаки (знаки операций, пунктуации и т.д.
множество текстов некоторого алфавита, удовлетворяющих правилам синтаксиса и задающих порядок вычислений в соответствии с правилами.
совокупность правил записи, которым должна удовлетворять программа, включает также правила ввода текстов программ в ЭВМ
#2/
Обладает состоянием, поведением и идентичностью
$
Объект
Класс
Свойство
Операция
#2/
То, как объект действует и реагирует называют
$
Поведением объекта
Свойством объекта
Методом объекта
Сообщением
#2/
Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию называется
$
Сообщением
Методом
Поведением
Свойством
#2/
Операции, выполняемые над данным объектом, называются
$
Методами
Свойствами
Поведением
Классом
#3/
Установите соответствие:
$
Модификатор-
Селектор-
Итератор-
Конструктор-
Деструктор-
$
операция, которая изменяет состояние объекта
операция, считывающая состояние объекта, но не меняющая состояние
операция, позволяющая организовать доступ ко всем частям объекта в строго определенной последовательности.
Операция создания и/или его инициализации
Операция, освобождающая состояние объекта и/или разрушающая сам объект
#2/

Некое множество объектов, имеющих общую структуру и общее поведение, называется
$
Классом
Полем
Типом
Идентичностью
Событием
#2/
Свойство классов решать схожие по смыслу проблемы разными способами называется
$
полиморфизмом
декомпозицией
инкапсуляцией
наследованием
декомпиляцией
#2/
Любой класс может быть порожден от другого класса. Это свойство называется
$
наследованием
декомпозицией
инкапсуляцией
полиморфизмом
заимствованием
#2/
Инкапсулированные в классе данные называются
$
полями
методами
свойствами
событиями
инкапсуляцией
декомпиляцией
#2/
Процесс связывания методов с объектами во время компиляции называется
$
Ранним связыванием
Поздним связыванием
Консервативным связыванием
Привязкой
#2/
Сразу после создания Windows программы взаимодействовали непосредственно с Windows API-интерфейсом (Application Programming Interface — программный интерфейс приложения), представляющим собой большой набор методов, которые могут вызываться программами для доступа
$
к функциям Windows.
к ресурсам Windows.
к программам Windows.
к интерфейсу Windows.
#3/
Установите соотвествие:
$
Консольное приложение
Оконное приложение
$
Именно ваша программа инициирует взаимодействие с операционной системой Windows. Программы, написанные таким способом, сами обращаются к операционной системе, а не операционная система к ним.
Именно Windows должна обращаться к вашей программе. Процесс взаимодействия организован следующим образом: программа ожидает до тех пор, пока не получит сообщение от Windows. Получив его, программа должна предпринять соответствующее действие.
Отвечая на сообщение, приложение не может вызвать метод, определенный в Windows, но главное здесь то, что инициатором взаимодействия все-таки является приложениеЮ посылающее сообщение . Windows
#2/
Содержит классы для создания приложений Windows, которые позволяют использовать возможности стандартного пользовательского интерфейса, доступные в операционной системе Microsoft Windows.

$
Пространство имен System.Windows.Forms
Пространство имен System.Windows
Пространство имен System.Console
Пространство имен System.Forms

Задания для оценки умений

1. Тест:

#2/
В качестве базового элемента объектно-ориентированного программирования используются
$
Объекты
Алгоритмы
Потоки
Подпрограммы
#2/
Неверным высказыванием является
$
класс является экземпляром объекта
объекты в процессе выполнения программы создаются и уничтожаются
классы организованы иерархически
термины - «экземпляр класса» и «объект» - взаимозаменяемы
#2/
Объект не обладает
$
модульностью
состоянием
поведением
идентичностью
#2/
Перечнем всех свойств (атрибутов) данного объекта и текущим значением каждого из этих свойств характеризуется
$
состояние
модульность
поведение
идентичность
#2/
Неверным высказыванием является
$
Объект может порождать любое количество классов
Класс - это некое множество объектов, имеющих общую структуру и общее поведение
Классы и объекты - это тесно связанные понятия
Классы статичны, т.е. все их особенности и содержание определены в процессе компиляции программы
#2/
Событие это - что-то, что может изменить
$
состояние объекта
модульность объекта
поведение объекта
идентичность объекта
#2/
Операция создания и/или инициализации объекта -
$
конструктор
модификатор
селектор
итератор
деструктор
#2/
Операция, освобождающая состояние объекта и/или разрушающая сам объект
$

деструктор
модификатор
селектор
итератор
конструктор
#2/
Окончанием существования любого объекта является момент его
$
уничтожения
описания
инициализации
создания
использования
#2/
Свойство классов решать схожие по смыслу проблемы различными способами называется
$
Полиморфизмом
Абстрагированием
Идентичностью
Инкапсуляцией
Иерархией
Модульностью
#2/
Инкапсулированные в класса данные называются
$
Полями
Методами
Свойствами
#2/
Программа действий над объектом или его свойствами называется
$
Методами
Полями
Свойствами
#2/
Выберите ошибочную запись доступа к свойствам и методам объектов:
$
имя свойства . имя объекта
имя объекта . имя свойства
имя объекта . имя метода
#2/
Окно формы
$
проект Windows- окна будущей программы
предназначено для отображения свойств и событий объекта
осуществляет основные функции управления проектом создаваемой программы
это палитра компонентов
предназначено для быстрого доступа к наиболее важным опциям главного меню
предназначено для создания и редактирования текста программы

#2/
Укажите ложное высказывание:
$
Процедуры нельзя использовать повторно. Можно создавать библиотеки однократно используемых процедур.
Процедурное программирование как методика разработки программного обеспечения разделяет программу на данные и процедуры, обрабатывающие эти данные.
Процедурное программирование имеет последовательную природу. В процессе выполнения процедурной программы последовательно вызываются различные процедуры. После выполнения последней из них программа завершает свою работу.
Процедурное программирование задает общую структуру программы: 1) данные 2) и процедуры.
Использование процедур помогает разрабатывать программу, предназначенную для решения конкретной задачи. Вместо написания одного большого блока обработки данных в ходе разработки программа все время разбивается на процедуры и более мелкие подпрограммы.
#2/

Укажите ложное высказывание:
$
Как процедурное, так и модульное программирование не поддерживают многократное использование.
Модульное программирование жестко связывает данные и процедуры, обрабатывающие эти данные, в компоненты программы, называемые модулями.
Модули скрывают представление данных и внутренние методы их обработки. Однако большинство модульных языков программирования дают возможность использовать модули в процедурной среде.
Модульное программирование скрывает детали реализации алгоритмов, защищая, таким образом, данные от несовместимого или некорректного изменения.
Использование модулей также задает высокоуровневую структуру программы. Такой подход дает возможность разрабатывать структуру программы на концептуальном, поведенческом уровне, а не в терминах данных и процедур.
#2/
Укажите ложное высказывание:
$
Объектно-ориентированное программирование является развитием модульного программирования, и как следствие, так же не поддерживает многократное использование.
Объектно-ориентированное программирование — метод разработки программного обеспечения, позволяющий при моделировании программы использовать названия объектов реального мира. Оно разбивает программу на набор взаимодействующих объектов.
Объектно-ориентированное программирование является развитием модульного программирования, так как поддерживает инкапсуляцию.
По сравнению с модульным программированием, в ООП введена поддержка таких свойств объектов, как наследование (для расширения возможности многократного использования) и полиморфизм (для повышения гибкости в работе с типами данных).
#2/
Укажите свойства, которыми обладают написанные с помощью объектно-ориентированного программирования программы:
$
Естественность, Надежность, Возможность многократного использования, Удобство в сопровождении (легкость исправления ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Естественность, Надежность, Возможность однократного использования, Удобство в сопровождении (легкость исправления ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Естественность, Надежность, Возможность многократного использования, Проблемы в сопровождении (тяжело исправлять ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
Научность, Надежность, Возможность многократного использования, Проблемы в сопровождении (тяжело исправлять ошибок), Расширяемость, Быстрота создания (своевременный периодический выпуск (издание) новых версий)
#2/
Объектно-ориентированное программирование естественно. Вместо того чтобы описывать задачу в терминах данных и процедур, при использовании объектно-ориентированного программирования можно пользоваться словарем исходной задачи. Такой подход, давая возможность использовать хорошо понятные термины, позволяет сосредоточить внимание
$
На решении задачи в терминах объектов и классов, а не на деталях реализации.
На деталях реализации, а не на решении задачи в терминах объектов и классов .
На определении процедур для обработки данных предметной области
На определении структуры модулей предметной области
#2/
Ключевое слово языка программирования, которое говорит компьютеру, какая часть программы может получить доступ к атрибутам и методам, являющимся членами класса это -
$
Спецификатор доступа
Слово NEW
Слово Constructor
Имя метода
#2/
Спецификатор доступа _____определяет атрибуты и методы, которые доступны при использовании экземпляра любого класса.
$
Public

Private
Protected
New
#2/
Спецификатор доступа _____ определяет атрибуты и методы, которые доступны только методам, определенным в данном классе.
$
Private
Public
Protected
New
#2/
Спецификатор доступа_____ определяет атрибуты и процедуры, которые могут быть наследованы другими классами (производными), но не видимы всеми классами.
$
Protected
Public
Private
New
#2/
Производный класс наследует -_____и _____члены базового класса.
$
Public и protected
Private и public
Private и protected
New и protected
#2/
На рисунке изображена иерархия классов
Какой из классов является абстрактным?

$
Shape
Oval
Hexagon
Circle
ShapesApp

#2/
На рисунке изображена иерархия классов
Какой из классов является производным от абстрактного класса?

$
Circle
Shape
Oval
ShapesApp
#2/
На рисунке изображена диаграмма классов. Укажите класс являющийся интерфейсом

$
IGameObject и IPlayer
Application
Smith
#2/
На рисунке изображена диаграмма классов. Укажите класс, использующий интерфейс

$
Smith
IGameObject и IPlayer
Application
#2/
Каждый язык программирования определяется
$
Алфавитом, синтаксисом, семантикой

Символами, словами и предложениями
Правилами организации и структурирования
#4/
Расставьте методологии программирования в соответствии с этапами развития технологий программирования
$
Методологии программирования нет, программирование считается искусством.
Методология - структурный подход.
Методология - модульный подход.
Методология - объектно-ориентированный подход.
#2/
Под семантикой языка программирования понимают
$
правила, определяющие какие операции, и в какой последовательности должна выполнять ЭВМ, работающая по программе, т.е. это смысловое значение слов и фраз написанных на языке программирования.

набор символов, включающий буквы, цифры и специальные знаки (знаки операций, пунктуации и т.д.
множество текстов некоторого алфавита, удовлетворяющих правилам синтаксиса и задающих порядок вычислений в соответствии с правилами.
совокупность правил записи, которым должна удовлетворять программа, включает также правила ввода текстов программ в ЭВМ
#2/
Обладает состоянием, поведением и идентичностью
$
Объект
Класс
Свойство
Операция
#2/
То, как объект действует и реагирует называют
$
Поведением объекта
Свойством объекта
Методом объекта
Сообщением
#2/
Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию называется
$
Сообщением
Методом
Поведением
Свойством
#2/
Операции, выполняемые над данным объектом, называются
$
Методами
Свойствами
Поведением
Классом
#3/
Установите соответствие:
$
Модификатор-
Селектор-
Итератор-
Конструктор-
Деструктор-
$
операция, которая изменяет состояние объекта
операция, считывающая состояние объекта, но не меняющая состояние
операция, позволяющая организовать доступ ко всем частям объекта в строго определенной последовательности.
Операция создания и/или его инициализации
Операция, освобождающая состояние объекта и/или разрушающая сам объект
#2/

Некое множество объектов, имеющих общую структуру и общее поведение, называется
$
Классом
Полем
Типом
Идентичностью
Событием
#2/
Свойство классов решать схожие по смыслу проблемы разными способами называется
$
полиморфизмом
декомпозицией
инкапсуляцией
наследованием
декомпиляцией
#2/
Любой класс может быть порожден от другого класса. Это свойство называется
$
наследованием
декомпозицией
инкапсуляцией
полиморфизмом
заимствованием
#2/
Инкапсулированные в классе данные называются
$
полями
методами
свойствами
событиями
инкапсуляцией
декомпиляцией
#2/
Процесс связывания методов с объектами во время компиляции называется
$
Ранним связыванием
Поздним связыванием
Консервативным связыванием
Привязкой
#2/
Сразу после создания Windows программы взаимодействовали непосредственно с Windows API-интерфейсом (Application Programming Interface — программный интерфейс приложения), представляющим собой большой набор методов, которые могут вызываться программами для доступа
$
к функциям Windows.
к ресурсам Windows.
к программам Windows.
к интерфейсу Windows.
#3/
Установите соотвествие:
$
Консольное приложение
Оконное приложение
$
Именно ваша программа инициирует взаимодействие с операционной системой Windows. Программы, написанные таким способом, сами обращаются к операционной системе, а не операционная система к ним.
Именно Windows должна обращаться к вашей программе. Процесс взаимодействия организован следующим образом: программа ожидает до тех пор, пока не получит сообщение от Windows. Получив его, программа должна предпринять соответствующее действие.
Отвечая на сообщение, приложение не может вызвать метод, определенный в Windows, но главное здесь то, что инициатором взаимодействия все-таки является приложениеЮ посылающее сообщение . Windows
#2/
Содержит классы для создания приложений Windows, которые позволяют использовать возможности стандартного пользовательского интерфейса, доступные в операционной системе Microsoft Windows.

$
Пространство имен System.Windows.Forms
Пространство имен System.Windows
Пространство имен System.Console
Пространство имен System.Forms

Задания для оценки владений

1. Задача:

Создать 2 формы для игры или ЦОР по любой теме школьного курса:
Анимационная заставка
Поле для настольной игры.
Формы должна быть «резиновыми». Для рисования поля использовать кисти с рисунками. Для рисования фишек или фигур различные градиентные заливки. В анимации должно участвовать не менее 3 различных фигур.
База данных хроники восхождений в альпинистском клубе.
В базе данных должны записываться даты начала и завершения каждого восхождения, имена и адреса участвовавших в нем альпинистов, название и высота горы, страна и район, где эта гора расположена. Дайте выразительные имена таблицам и полям, в
которые могла бы заноситься указанная информация. Написать
пакет, состоящий из процедур и функций, которые позволил бы
выполнить следующие действия с базой данных:
1) для каждой горы показать список групп, осуществлявших
восхождение, в хронологическом порядке;
2) предоставить возможность добавления новой вершины, с
указанием названия вершины, высоты и страны местоположения;
3) предоставить возможность изменения данных о вершине, если на нее не было восхождения;
4) показать список альпинистов, осуществлявших восхождение
в указанный интервал дат;
3.1. Варианты заданий
5) предоставить возможность добавления нового альпиниста в состав указанной группы;
6) показать информацию о количестве восхождений каждого
альпиниста на каждую гору;
7) показать список восхождений (групп), которые осуществлялись в указанный пользователем период времени;
8) предоставить возможность добавления новой группы, указав
ее название, вершину, время начала восхождения;
9) предоставить информацию о том, сколько альпинистов побывали на каждой горе.

Раздел: Реализация принципов ООП в C#. Визуальное программирование.

Задания для оценки знаний

1. Тест:

#3/
Установите соответствие:
$
Меню и панели инструментов
Элементы управления
Макет
Компоненты
$
ToolStrip,MenuStrip,ContextMenuStrip и StatusStrip
TextBox, ComboBox, Label, ListView, Button, WebBrowser
FlowLayoutPanel TableLayoutPanel SplitContainer
ToolTip, ErrorProvider, Help, HelpProvider
OpenFileDialog и SaveFileDialog FontDialog, PageSetupDialog, PrintPreviewDialog и PrintDialog
#2/
Класс Application
Метод запускает цикл обработки сообщений приложения в текущем потоке и, при необходимости, делает главную форму видимой.
$

Run
Exit
DoEvents
AddMessageFilter
Show
#2/
Класс Form
Название формы в проекте. Это не заголовок формы, который вы видите при запуске формы, а название формы внутри проекта, которое вы будете использовать в коде
$
Name
AcceptButton
BackColor
BackgroundImage
Text
Caption
#2/
Класс Form
Свойство ControlBox
$
Устанавливается наличие либо отсутствие трех стандартных кнопок в верхнем правом углу формы: "Свернуть", "Развернуть" и "Закрыть"
Устанавливается значение кнопки, которая будет срабатывать при нажатии клавиши Enter. Для того чтобы это свойство было активным, необходимо наличие по крайней мере одной кнопки, расположенной на форме
Цвет формы.
Устанавливается значение кнопки, которая будет срабатывать при нажатии клавиши Esc. Для того чтобы это свойство было активным, необходимо наличие по крайней мере одной кнопки, расположенной на форме
#3/
Установите соответствие
$
Cursor
Font
FormBorderStyle
Size
$
Определяется вид курсора при его положении на форме
Форматирование шрифта, используемого для отображения текста на форме в элементах управления
Определение вида границ формы.
Ширина и высота формы
#3/
Установите соответствие
$
KeyDown
KeyPress
KeyUp
$
Происходит при нажатии клавиши, если элемент управления имеет фокус
Происходит при нажатии клавиши, если элемент управления имеет фокус
Происходит, когда отпускается клавиша, если элемент управления имеет фокус
#3/
Установите соответствие
$
MouseClick
MouseDoubleClick
MouseDown
MouseEnter
MouseLeave
MouseMove
MouseUp
$
Генерируется при щелчке элемента управления мышью
Генерируется при двойном щелчке элемента управления мышью
Происходит при нажатии кнопки мыши, если указатель мыши находится на элементе управления
Происходит, когда указатель мыши оказывается на элементе управления

Происходит, когда указатель мыши покидает элемент управления
Происходит при перемещении указателя мыши по элементу управления
Происходит при отпускании кнопки мыши, когда указатель мыши находится на элементе управления
Генерируется при движении колесика мыши, если элемент управления имеет фокус
#2/
Метод Activate
$
Активирует форму и переводит на нее фокус.
Закрывает форму.
Скрывает элемент управления от пользователя
Отображает элемент управления для пользователя.
#2/
Метод Close
$
Закрывает форму.
Активирует форму и переводит на нее фокус.
Скрывает элемент управления от пользователя
Отображает элемент управления для пользователя.
#2/
Метод Hide
$
Скрывает элемент управления от пользователя
Активирует форму и переводит на нее фокус.
Закрывает форму.
Отображает элемент управления для пользователя.
#2/
Метод Show()
$
Отображает элемент управления для пользователя.
Активирует форму и переводит на нее фокус.
Закрывает форму.
Скрывает элемент управления от пользователя
#2/
private void button_Click(object sender, System.EventArgs e)
{
Обработчик события
$
Щелчок мышью по кнопке
Щелчок мышью по форме
Нажатие кнопки на клавиатуре
Перемещение мыши над кнопкой
Щелчок кнопкой по мыши
#2/
Событие KeyPress
$
которое может возникать несколько раз, когда пользователь удерживает нажатую клавишу
которое может возникать только один раз, когда пользователь удерживает нажатую клавишу
которое возникает один раз после того, как пользователь отпускает клавишу
#3/
Установите соответствие:
$
LABEL
LinkLabel
TextBox
RichTextBox
$
чаще всего используется для информирования пользователя. На неё заносится текст
позволяет добавлять гиперссылки в приложения.
Используется для ввода и редактирования неформатированного текста пользователем.
Используется для форматированных текстов.
Предлагает список объектов, из которого можно выбрать нужные.
позволяет перемещаться по числовому ряду в определённом диапазоне
#2/
На рисунке изображен компонент:

$
NumericUpDown
ToolTip
CheckListBox
ComboBox
ListBox
MaskedTextBox
#2/
На рисунке изображен компонент:

$
ListBox
NumericUpDown
ToolTip
CheckListBox
ComboBox
MaskedTextBox
#2/
На рисунке изображен компонент:

$
CheckListBox
ListBox
NumericUpDown
ToolTip
ComboBox
MaskedTextBox
#2/
На рисунке изображен компонент:

$
ComboBox
ListBox
NumericUpDown
ToolTip
CheckListBox
MaskedTextBox
#2/
MaskedTextBox
$
Предназначен для ввода текста с заранее заданным или стандартным форматом, например, телефонные номера, IP адреса, даты и тп.
Создаёт всплывающее окно, со сведениями об элементе управления в интерфейсе, на который наведена мышка.
Производит визуальную демонстрацию протекания процесса.
Используется для форматированных текстов.
чаще всего используется для информирования пользователя. На неё заносится текст
позволяет добавлять гиперссылки в приложения.
#2/
На рисунке изображен компонент:

$
MonthCalendar
DateTimePicker
Timer
DateTime
Calendar
#2/
На рисунке изображен компонент:

$
DateTimePicker
MonthCalendar
Timer

DateTime
Calendar
#2/
Двухмерная векторная графика
$
включает в себя примитивы (прямые и кривые линии, геометрические фигуры), заданные набором точек в системе координат
связана с отображением текста на экране путем использования различных шрифтов, размеров и стилей
хранится в виде точечных рисунков — массивов чисел, каждое из которых представляет цвет определенной точки на рисунке
#2/
Типографская разметка
$
связана с отображением текста на экране путем использования различных шрифтов, размеров и стилей
включает в себя примитивы (прямые и кривые линии, геометрические фигуры), заданные набором точек в системе координат
хранится в виде точечных рисунков — массивов чисел, каждое из которых представляет цвет определенной точки на рисунке
#2/
Класс используется для рисования линий и кривых,
$
Pen
Brush
Bitmap
Region
Color
#2/
классы, производные от абстрактного класса используются для заливки фигур.
$
Brush
Pen
Bitmap
Region
Color
#2/
Метод DrawRectangle
$
Рисуем прямоугольник
Рисуем закрашенный прямоугольник
Рисуем произвольный многоугольник
Рисуем произвольный закрашенный многоугольник
#2/
Определяет кисть одного цвета.
$
SolidBrush,
TextureBrush
LinearGradientBrush.
#4/
Расставьте операторы в верной последовательности
$
SolidBrush brush1 =
new SolidBrush(Color.DarkOrchid);
e.Graphics.FillRectangle (brush1,
new Rectangle (200, 200, 250, 230));
brush1.Dispose();
#4/
Расставьте операторы в верной последовательности
$
SolidBrush myBrush =
new SolidBrush(Color.Red);
e.Graphics.FillEllipse(myBrush,
new Rectangle(0, 0, 200, 300));
myBrush.Dispose();
#2/

```
String drawString = "Sample Text";
Font drawFont = new Font("Arial", 16);
SolidBrush drawBrush = new SolidBrush(Color.Black);
float x = 150.0F;
float y = 150.0F;
float width = 200.0F;
 float height = 50.0F;
RectangleF drawRect = new RectangleF(x, y, width, height);
e.Graphics.DrawString(drawString, drawFont, drawBrush, drawRect);
```
$
Вывод текста в прямоугольную область
Вывод текста с точки
Вывод текста по вертикали
Рисование прямоугольника

#2/
Набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени.
$
Надежность (Reliability)
Функциональные возможности (Functionality)
Практичность (Usability)
Эффективность (Efficiences)
Сопровождаемость (Maintainability)
Мобильность (Portability)
#2/
Набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений
$
Сопровождаемость (Maintainability)
Надежность (Reliability)
Функциональные возможности (Functionality)
Практичность (Usability)
Эффективность (Efficiences)
Мобильность (Portability)
#2/
Набор атрибутов, относящихся к объему работ, требуемых для использования и индивидуальной оценки такого использования определенным или предполагаемым кругом пользователей
$
Практичность (Usability)
Надежность (Reliability)
Функциональные возможности (Functionality)
Эффективность (Efficiences)
Сопровождаемость (Maintainability)
Мобильность (Portability)
#2/
Набор атрибутов, относящихся к способности программного обеспечения быть перенесенным из одного окружения в другое.
$
Мобильность (Portability)
Надежность (Reliability)
Функциональные возможности (Functionality)
Практичность (Usability)
Эффективность (Efficiences)
Сопровождаемость (Maintainability)
#3/
Установите соответствие:
$
Пригодность (Suitability)
Правильность (Accuracy)
Способность к взаимодействию (Interoperability)
Согласованность (Compliance)
Защищенность (Security)
$
Атрибут ПО, относящийся к наличию и соответствию набора функций конкретным задачам.

Атрибуты ПО, относящиеся к обеспечению соответствия результатов или эффектов.
Атрибуты ПО, относящиеся к способности его взаимодействовать с конкретными системами.
Атрибуты ПО, которые заставляют программу придерживаться соответствующих стандартов или соглашений, или положений законов, или подобных рекомендаций.
Атрибуты ПО, относящиеся к его способности предотвращать несанкционированный доступ, случайный или преднамеренный, к программам и данным.
#3/
Установите соответствие:
$
Стабильность (Maturity)
Устойчивость к ошибке (Fault tolerance)
Восстанавливаемость (Recoverability)
$
Атрибуты программного обеспечения, относящиеся к частоте отказов при ошибках в программном обеспечении.
Атрибуты программного обеспечения, относящиеся к его способности поддерживать определенный уровень качества функционирования в случаях программных ошибок или нарушения определенного интерфейса.
Атрибуты программного обеспечения, относящиеся к его возможности восстанавливать уровень качества функционирования и восстанавливать данные, непосредственно поврежденные в случае отказа, а также к времени и усилиям, необходимым для этого.

Задания для оценки умений

1. Задача:

Анимационная заставка
Поле для настольной игры.
Формы должна быть «резиновыми». Для рисования поля использовать кисти с рисунками. Для рисования фишек или фигур различные градиентные заливки. В анимации должно участвовать не менее 3 различных фигур.
База данных хроники восхождений в альпинистском клубе.
В базе данных должны записываться даты начала и завершения каждого восхождения, имена и адреса участвовавших в нем альпинистов, название и высота горы, страна и район, где эта гора
расположена. Дайте выразительные имена таблицам и полям, в
которые могла бы заноситься указанная информация. Написать
пакет, состоящий из процедур и функций, которые позволил бы
выполнить следующие действия с базой данных:
1) для каждой горы показать список групп, осуществлявших
восхождение, в хронологическом порядке;
2) предоставить возможность добавления новой вершины, с
указанием названия вершины, высоты и страны местоположения;
3) предоставить возможность изменения данных о вершине, если на нее не было восхождения;
4) показать список альпинистов, осуществлявших восхождение
в указанный интервал дат;
3.1. Варианты заданий
5) предоставить возможность добавления нового альпиниста в состав указанной группы;
6) показать информацию о количестве восхождений каждого
альпиниста на каждую гору;
7) показать список восхождений (групп), которые осуществлялись в указанный пользователем период времени;
8) предоставить возможность добавления новой группы, указав
ее название, вершину, время начала восхождения;
9) предоставить информацию о том, сколько альпинистов побывали на каждой горе.
Предусмотреть разработку триггер

Задания для оценки владений

1. Задача:

Анимационная заставка
Поле для настольной игры.
Формы должна быть «резиновыми». Для рисования поля использовать кисти с рисунками. Для рисования фишек или фигур различные градиентные заливки. В анимации должно участвовать не менее 3 различных фигур.
База данных хроники восхождений в альпинистском клубе.

В базе данных должны записываться даты начала и завершения каждого восхождения, имена и адреса участвовавших в нем альпинистов, название и высота горы, страна и район, где эта гора расположена. Дайте выразительные имена таблицам и полям, в которые могла бы заноситься указанная информация. Написать пакет, состоящий из процедур и функций, которые позволил бы выполнить следующие действия с базой данных:
1) для каждой горы показать список групп, осуществлявших восхождение, в хронологическом порядке;
2) предоставить возможность добавления новой вершины, с указанием названия вершины, высоты и страны местоположения;
3) предоставить возможность изменения данных о вершине, если на нее не было восхождения;
4) показать список альпинистов, осуществлявших восхождение в указанный интервал дат;
3.1. Варианты заданий
5) предоставить возможность добавления нового альпиниста в состав указанной группы;
6) показать информацию о количестве восхождений каждого альпиниста на каждую гору;
7) показать список восхождений (групп), которые осуществлялись в указанный пользователем период времени;
8) предоставить возможность добавления новой группы, указав ее название, вершину, время начала восхождения;
9) предоставить информацию о том, сколько альпинистов побывали на каждой горе.
Предусмотреть разработку триггер

2. Оценочные средства для промежуточной аттестации

1. Экзамен

Вопросы к экзамену:

1. the common language runtime (CLR)
2. the .NET Framework class library (.NET FCL).
3. Среда Времени Выполнения или Виртуальная Машина.
4. .NET Framework Class Library
5. MSIL (Microsoft Intermediate Language, он же IL – Intermedia Language) – промежуточный язык
6. JIT-компилятор
7. CTS - Common Type System Стандартная Система Типов.
8. Сборка мусора
9. Пространство имён
10. Структура проекта консольного приложения
11. Типы данных языка. Типы значений и ссылочные типы
12. Какие идентификаторы и диапазоны значений имеют числовые типы данных?
13. Какие операции можно осуществлять над переменными целого и вещественного типов?
14. Каков формат записи операции присваивания?
15. Какие встроенные функции существуют для работы с переменными целых и вещественных типов? Что необходимо сделать, чтобы использовать их в программе?
16. Как выполняется преобразование типов в выражениях?
17. Как выполняется ввод и вывод данных на экран в C#?
18. Каков приоритет выполнения операций в C#?
19. Что представляет собой логический тип данных в языке C#?
20. Как записываются логические операции в C#?
21. Как записывается условный оператор в языке C#?
22. Каковы правила записи составных условий в C#?
23. Какие виды циклов определены в языке C#?
24. Для каждого вида цикла ответьте на следующие вопросы:
25. В каких ситуациях удобно использовать тот или иной оператор цикла?
26. Какова структура цикла?
27. Объясните принцип работы цикла. Каково условие выхода из цикла?
28. Как записывается оператор выбора в языке C#?
29. Какой идентификатор имеет перечисляемый тип в языке C#? Как он записывается?
30. Что может быть значением в перечисляемом типе?

31. Класс Random и его функции
32. Класс Array: методы и свойства (из справки в лабораторной работе и примеров лекций)
33. Создание линейного массива, работа с элементами массива.
34. Цикл foreach
35. Примеры программ на линейные массивы и матрицы (типа задач ЕГЭ, примеры http://inf.reshuege.ru/test?theme=284)
36. Матрицы (примеры тестов в лабораторной работе и из примеров выше)
37. Класс string: методы и свойства (из справки в лабораторной работе и примеров лекций)
38. Объявление класса
39. Методы. Параметры методов (Примеры тестов в лабораторной работе)
40. Что такое процедурное программирование?
41. Какими преимуществами обладает процедурное программирование по сравнению с неструктурным программированием?
42. Что такое модульное программирование?
43. Какими преимуществами обладает модульное программирование по сравнению с процедурным программированием?
44. Перечислите недостатки процедурного и модульного программирования.
45. Что такое объектно-ориентированное программирование?
46. Каковы шесть преимуществ и целей объектно-ориентированного программирования?
47. Объясните одну из целей объектно-ориентированного программирования.
48. Дайте определения следующих терминов:класс и объект
49. Как объекты обмениваются информацией?
50. Что такое конструктор?
51. Что такое средство доступа?
52. Что такое this?
53. Что такое переменная экземпляра?
54. Как определить размер класса?
55. Объясните шаги, необходимые для создания экземпляра класса.
56. Как объявляются переменные экземпляра?
57. Что такое тип данных?
58. Объясните части определения метода.
59. Какая разница между списком аргументов и списком параметров?
60. Объясните шаги, необходимые для вызова метода-члена.
61. Что такое конструктор и как он определяется?
62. Как обращаться к переменным экземпляра из программы?
63. Что такое инкапсуляция?
64. Каковы преимущества использования инкапсуляции?
65. Что такое спецификатор(модификатор) доступа?
66. Что такое спецификатор доступа public?
67. Что такое спецификатор доступа private?
68. Что такое спецификатор доступа protected?
69. Какие части базового класса могут использоваться производным классом?
70. Почему программисты требуют, чтобы доступ к некоторым атрибутам класса осуществлялся только с помощью методов-членов?
71. Каким образом использование инкапсуляции помогает достичь целей объектно-ориентированного программирования?
72. Дайте определение понятию "абстракция" и приведите пример применения абстракции.
73. Дайте определение понятию "реализация".
74. Дайте определение понятию "интерфейс".
75. Объясните разницу между интерфейсом и реализацией.
76. Почему для достижения эффективной инкапсуляции важно четко распределить ответственность?
77. Определите понятие типа.
78. Что такое абстрактный тип данных?
79. Назовите два типа конструкторов.
80. Какие недостатки имеет простое многократное использование?
81. Что такое наследование?
82. Назовите три формы наследования.
83. В чем состоит опасность, когда применяется наследование?
84. Что такое программирование отличий?

85. Когда класс-наследник может иметь три типа методов и свойств? Что это за типы? В чем состоит выгода от применения программирования отличий?
86. Какие классы являются потомками класса Permission (пример в лекциях)?
87. Корневой класс и классы-листья.
88. Что такое наследование для заменяемости типов?
89. Как наследование разрушает инкапсуляцию? Как применить инкапсуляцию при наследовании
90. Что такое полиморфизм?
91. Как реализуется полиморфизм?
92. Что такое позднее связывание?
93. Что такое раннее связывание?
94. Каковы преимущества полиморфизма времени выполнения?
95. Каковы преимущества полиморфизма времени компиляции?
96. Что такое интерфейс?
97. Как полиморфизм позволяет реализовать интерфейсы?
98. Что такое виртуальная функция?
99. Что такое перегрузка метода?
100. Что такое интерфейс?
101. Чем интерфейс отличается от абстрактного класса?
102. С какими недостатками языка справляются интерфейсы?
103. В чем состоит основное назначение интерфейсов?
104. Стандартные интерфейсы C#
105. Пространство имен System.Collections
106. Операции is и as
107. Как Windows взаимодействует с пользователем?
108. Форма инкапсулирует основные функции, необходимые для чего?
109. Класс Control
110. Важные свойства объекта Control
111. Важные методы объекта Control
112. Важные методы объекта Control
113. Важные события класс Control
114. Дополнительные события класса Control
115. Передача параметров между окнами
116. Окна родитель и потомок
117. Языковая кастомизация
118. Значок в трее – NotifyIcon
119. Процессы работы с данными:
120. Основы ADO .NET
121. Принцип единообразной работы с базами данных
122. Доступ к отсоединенным данным
123. ADO .NET. Доступ к данным
124. Набор данных, провайдер данных
125. ADO .NET. Объектная модель
126. Классы подсоединенных объектов
127. Классы отсоединенных объектов
128. объект DataTable
129. свойство Relations
130. класс Constraint
131. объект DataSet
132. Платформа .NET Framework

Практические задания:

1. У вас есть доллары. Вы хотите обменять их на рубли. Есть информация о стоимости купли-продажи в банках города. В городе N банков. Составьте про¬грамму, определяющую, какой банк выбрать, чтобы выгодно обменять доллары на рубли. Какой банк выбрать, если необходимо купить доллары?
2. Даны результаты соревнования по бегу (время в секундах). Вывести на экран результаты первой тройки (использовать сортировку массива).
3. Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести инициалы и фамилию.

4. Тестирование коллектива. Пусть матрица размером N x M содержит информацию о результатах тестирования студентов некоторой подгруппы из N человек. В j-ом столбце каждого студента стоит 1, если он правильно ответил на j-ое тестовое задание, и 0 – в противном случае. Всего M тестовых заданий.

2. Курсовая работа

Темы курсовых работ:

1. Разработка программы-тренажера по развитию навыков работы с компьютерной мышью
2. Разработка интерактивной игры по развитию логики и внимания
3. Программная реализация алгоритма поиска пути в лабиринте
4. Создание интерактивных приложений средствами платформы Unity
5. Программная демонстрация методов кодирования растровых графических изображений
6. Разработка интерактивных физических игр средствами Unity
7. Разработка интерактивных математических игр
8. Разработка чат-бота для учебного центра
9. Разработка образовательной игры "Графы"
10. Разработка приложения для игры в ребусы
11. Разработка игрового приложения "Головоломки"
12. Алгоритмы закрашивания геометрических фигур
13. Разработка сетевого чата для системного администратора
14. Разработка демонстрационной программы по помехоустойчивому кодированию
15. Реализация алгоритмов компьютерной алгебры
16. Программирование аналоговых датчиков аппаратно-программной системы Arduino
17. Программирование цифровых датчиков аппаратно-программной системы Arduino
18. Программирование цифровых датчиков аппаратно-программной системы Arduino с помощью графической среды FLProg
19. Создание интерактивныой игры по изучению основных терминов информатики на Unity 3D
20. Создание интерактивныой игры по изучению инространных слов на Unity 3D
21. Разработка программы-тренажера по информатике по теме "Логика"

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

1. Для текущего контроля используются следующие оценочные средства:

1. Задача

Задачи позволяют оценивать и диагностировать знание фактического материала (базовые понятия, алгоритмы, факты) и умение правильно использовать специальные термины и понятия, узнавание объектов изучения в рамках определенного раздела дисциплины;
умения синтезировать, анализировать, обобщать фактический и теоретический материал с формулированием конкретных выводов, установлением причинно-следственных связей.
Алгоритм решения задач:
1.Внимательно прочитайте условие задания и уясните основной вопрос, представьте процессы и явления, описанные в условии.
2.Повторно прочтите условие для того, чтобы чётко представить основной вопрос, проблему, цель решения, заданные величины, опираясь на которые можно вести поиск решения.
3.Произведите краткую запись условия задания.
4.Если необходимо, составьте таблицу, схему, рисунок или чертёж.
5.Установите связь между искомыми величинами и данными; определите метод решения задания, составьте план решения.
6.Выполните план решения, обосновывая каждое действие.
7.Проверьте правильность решения задания.
8.Произведите оценку реальности полученного решения.
9.Запишите ответ.

2. Тест

Тест это система стандартизированных вопросов (заданий), позволяющих автоматизировать процедуру измерения уровня знаний и умений обучающихся. Тесты могут быть аудиторными и внеаудиторными. Преподаватель доводит до сведения студентов информацию о проведении теста, его форме, а также о разделе (теме) дисциплины, выносимой на тестирование.
При самостоятельной подготовке к тестированию студенту необходимо:
− проработать информационный материал по дисциплине. Проконсультироваться с преподавателем по вопросу выбора учебной литературы;
− выяснить все условия тестирования заранее. Необходимо знать, сколько тестов вам будет предложено, сколько времени отводится на тестирование, какова система оценки результатов и т.д.
− работая с тестами, внимательно и до конца прочесть вопрос и предлагаемые варианты ответов; выбрать правильные (их может быть несколько); на отдельном листке ответов выписать цифру вопроса и буквы, соответствующие правильным ответам. В случае компьютерного тестирования указать ответ в соответствующем поле (полях);
− в процессе решения желательно применять несколько подходов в решении задания. Это позволяет максимально гибко оперировать методами решения, находя каждый раз оптимальный вариант.
− решить в первую очередь задания, не вызывающие трудностей, к трудному вопросу вернуться в конце.
− оставить время для проверки ответов, чтобы избежать механических ошибок.

2. Описание процедуры промежуточной аттестации

Оценка за зачет/экзамен может быть выставлена по результатам текущего рейтинга. Текущий рейтинг – это результаты выполнения практических работ в ходе обучения, контрольных работ, выполнения заданий к лекциям (при наличии) и др. видов заданий.

Результаты текущего рейтинга доводятся до студентов до начала экзаменационной сессии.

Курсовая работа — студенческое научное исследование по одной из базовых дисциплин учебного плана либо специальности, важный этап в подготовке к написанию выпускной квалификационной работы. Темы работ предлагаются и утверждаются кафедрой. Студент может предложить тему самостоятельно, однако она не должна выходить за рамки учебного плана. На 1-2 курсах данная работа носит скорее реферативный характер, на старших – исследовательский. Работа обычно состоит из теоретической части (последовательное изложение подходов, мнений, сложившихся в науке по избранному вопросу) и аналитической (анализ проблемы на примере конкретной ситуации (на примере группы людей, организации). Объем курсовой работы составляет 20-60 страниц. По завершению работы над курсовой, студенты защищают ее публично перед своими однокурсниками и преподавателями.

Этапы выполнения курсовой работы:

1. выбор темы и ее согласование с научным руководителем;
2. сбор материалов, необходимых для выполнения курсовой работы;
3. разработка плана курсовой работы и его утверждение научным руководителем;
4. систематизация и обработка отобранного материала по каждому из разделов работы или проблеме с применением современных методов;
5. формулирование выводов и обсуждение их с научным руководителем;
6. написание работы согласно требованиям стандарта и методическим указаниям к ее выполнению (введение, главы основной части, заключение, приложения, список литературы).

При оформлении курсовой работы следует придерживаться рекомендаций, представленных в документе «Регламент оформления письменных работ».

Экзамен преследует цель оценить работу обучающегося за определенный курс: полученные теоретические знания, их прочность, развитие логического и творческого мышления, приобретение навыков самостоятельной работы, умения анализировать и синтезировать полученные знания и применять их для решения практических задач.

Экзамен проводится в устной или письменной форме по билетам, утвержденным заведующим кафедрой (или в форме компьютерного тестирования). Экзаменационный билет включает в себя два вопроса и задачи. Формулировка вопросов совпадает с формулировкой перечня вопросов, доведенного до сведения обучающихся не позднее чем за один месяц до экзаменационной сессии.

В процессе подготовки к экзамену организована предэкзаменационная консультация для всех учебных групп.

При любой форме проведения экзаменов по билетам экзаменатору предоставляется право задавать студентам дополнительные вопросы, задачи и примеры по программе данной дисциплины. Дополнительные вопросы также, как и основные вопросы билета, требуют развернутого ответа.