

УТВЕРЖДАЮ
Декан факультета

(подпись) Матвеев П.В.
ФИО
«___» _____ 20__

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ПРОГРАММИРОВАНИЕ

Направление/специальность подготовки	09.03.01 Информатика и вычислительная техника
Специализация/профиль/программа подготовки	Автоматизированные системы обработки информации и управления
Уровень высшего образования	Бакалавриат
Форма обучения	Очная
Факультет	И Информационных и управляющих систем
Выпускающая кафедра	И9 СИСТЕМ УПРАВЛЕНИЯ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Кафедра-разработчик рабочей программы	О7 Информационные системы и программная инженерия

КУРС	СЕМЕСТР	ОБЩАЯ ТРУДОЁМКОСТЬ (ЗАЧЕТНЫХ ЕДИНИЦ)	ЧАСЫ (по наличию видов занятий)									ВИД ПРОМЕЖУТОЧНОГО КОНТРОЛЯ
			ОБЩАЯ ТРУДОЁМКОСТЬ	АУДИТОРНЫЕ ЗАНЯТИЯ				САМОСТОЯТЕЛЬНАЯ РАБОТА				
				ВСЕГО	ЛЕКЦИИ	ЛАБОРАТОРНЫЙ ПРАКТИКУМ	ПРАКТИЧЕСКИЕ ЗАНЯТИЯ	ВСЕГО	КУРСОВОЙ ПРОЕКТ	КУРСОВАЯ РАБОТА	ДРУГИЕ ВИДЫ САМОСТ. РАБОТЫ	
1	2	4	144	68	34	0	34	76	0	0	76	диф. зач.
2	3	4	144	68	34	0	34	76	0	18	58	диф. зач.
ВСЕГО		8	288	136	68	0	68	152	0	18	134	

ЛИСТ СОГЛАСОВАНИЯ

**РАБОЧАЯ ПРОГРАММА СОСТАВЛЕНА В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ ФЕДЕРАЛЬНОГО
ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО ОБРАЗОВАНИЯ (ФГОС ВО)**

09.03.01 Информатика и вычислительная техника

год набора группы: 2024

Программу составил:

Кафедра О7 Информационные системы и программная инженерия
Вальштейн Константин Владимирович, старший преподаватель

Программа рассмотрена
на заседании кафедры-разработчика
рабочей программы **О7 Информационные системы и программная инженерия**

Заведующий кафедрой Семенова Е.Г., д.т.н., проф.

Программа рассмотрена
на заседании выпускающей кафедры

И9 СИСТЕМ УПРАВЛЕНИЯ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Заведующий кафедрой Матвеев С.А., к.т.н., доц.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ПРОГРАММИРОВАНИЕ

Разделы рабочей программы

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ
2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВО
3. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ
4. ФОРМЫ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ
5. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ
6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Приложения к рабочей программе дисциплины

- Приложение 1. Аннотация рабочей программы
- Приложение 2. Технологии и формы обучения
- Приложение 3. Фонды оценочных средств

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины является формирование следующих компетенций:

ПСК-1.1 — способность разрабатывать требования и проектировать программное обеспечение
ОПК-2 — способность понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности
ОПК-8 — способность разрабатывать алгоритмы и программы, пригодные для практического применения
ОПК-9 — способность осваивать методики использования программных средств для решения практических задач

Формированию компетенций служит достижение следующих результатов образования:

ПСК-1.1

знания:

Способов практической реализации программных проектов на языке C++;;;

умения:

Разрабатывать структуру программных приложений;;;

навыки:

Использовать распространенные методики разработки программного обеспечения;;.

ОПК-2

знания:

основы и техника обобщенного программирования;;;

основы и техника объектно-ориентированного программирования;;;

умения:

выбирать способы создания программных продуктов исходя из доступных языковых и инструментальных средств;;;

навыки:

программирования с использованием сторонних библиотек, расширяющих возможности базового языка программирования;;;

программирования статических и движущихся двумерных изображений на растровых устройствах отображения;;.

ОПК-8

знания:

понятия класса и объекта, понятия наследования, инкапсуляции и полиморфизма, понятие интерфейса класса и интерфейса библиотеки, виртуальные функции и их использование, базовые принципы функционирования программ с использованием библиотек семейств Simple DirectMedia Layer 2.x;;;

умения:

применять объектно-ориентированный подход при создании программных продуктов;;;

навыки:

программирования с использованием виртуальных функций и шаблонов;;;

программирования с использованием сторонних библиотек, расширяющих возможности базового языка программирования;;;

программирования статических и движущихся двумерных изображений на растровых устройствах отображения;;;

создания однооконных и многооконных Win32-приложений;;.

ОПК-9

знания:

методик объектно-ориентированного и обобщенного программирования;;;

умения:

разрабатывать Windows-приложения на языке C++;;;

разрабатывать Windows-приложения на языке C#;;;

навыки:

программирования с использованием виртуальных функций и шаблонов;;;

программирования с использованием сторонних библиотек, расширяющих возможности базового языка программирования;;;

программирования статических и движущихся двумерных изображений на растровых устройствах отображения;;;

создания однооконных и многооконных Win32-приложений;;.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВО

Дисциплина **ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ПРОГРАММИРОВАНИЕ** является дисциплиной **обязательной части блока 1** программы подготовки по направлению *09.03.01 Информатика и вычислительная техника*.

Содержание дисциплины является логическим продолжением дисциплин: **ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ИНФОРМАТИКА: ОСНОВЫ ПРОГРАММИРОВАНИЯ.**

Содержание дисциплины является основой для освоения дисциплин: **СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, БАЗЫ ДАННЫХ, ВЫПОЛНЕНИЕ И ЗАЩИТА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ.**

Предварительные компетенции, сформированные у обучающегося до начала изучения дисциплины:

- ОПК-2 — Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности
- ОПК-8 — Способен разрабатывать алгоритмы и программы, пригодные для практического применения
- ПК-91 — способен к коммуникации и кооперации в цифровой среде, использованию различных цифровых средств, позволяющих во взаимодействии с другими людьми достигать поставленных целей
- ПК-94 — способен к управлению информацией и данными, поиску источников информации и данных, восприятию, анализу, запоминанию и передаче информации с использованием цифровых средств, а также с помощью алгоритмов при работе с полученными из различных источников данными с целью эффективного использования полученной информации для решения задач

3. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины составляет 8 з.е., 288 ч.

3.1. Содержание (дидактика) дисциплины

КУРС	СЕМЕСТР	Наименование разделов и дидактических единиц	ВСЕГО	Аудиторные занятия в контактной форме			Самостоятельная работа студентов	Формируемая компетенция, %			
				ВСЕГО	Лекции	Практические занятия		ПСК-1.1	ОПК-2	ОПК-8	ОПК-9
1	2	Раздел 1. Основные понятия объектно-ориентированного программирования (ООП). 1.1. Парадигма ООП: инкапсуляция, наследование, полиморфизм. 1.2. Язык программирования (ЯП) C++ как язык, поддерживающий ООП. 1.3. Сравнение ЯП C++ и C. Указатели и ссылки. Возможность использования фрагментов текстов программ на C в программах на C++. 1.4. Обзор стандартной библиотеки C++. Пространства имен как механизм высокоуровневой инкапсуляции. Стандартные пространства имен. Заголовочные файлы стандартной библиотеки. 1.5. Поточковый ввод-вывод в C++. 1.6. Размещение динамических переменных. Операторы new, new[], delete, delete[]. 1.7. Многомерные динамические массивы. 1.8. Операции преобразование типа в C и C++. Операции static_cast, dynamic_cast, const_cast, reinterpret_cast.	9	4	4	0	5	10	10	10	10
1	2	Раздел 2. Стандартные и пользовательские типы данных в C++. Обработка исключений. Инкапсуляция и статический полиморфизм в C++. 2.1. Классы. Члены класса. Области видимости членов класса. 2.2. Конструкторы и деструкторы. Работа с экземпляром класса. Указатель this. 2.3. Исключения. Обработка исключений. Генерация исключений. Стандартные классы исключений. 2.4. Функции с параметрами по умолчанию. Перегрузка функций. 2.5. Конструктор копирования и оператор присваивания. Понятие статического полиморфизма. 2.6. Представление операций для классов. Операторные функции. 2.7. Дружественные функции и дружественные классы. 2.8. Статические члены класса.	26	12	6	6	14	5	5	5	5
1	2	Раздел 3. Наследование и динамический полиморфизм в C++. 3.1. Наследование. Иерархия классов. 3.2. Виртуальные функции. Понятие динамического полиморфизма. Виртуальные деструкторы. Оператор typeid. 3.3. Чисто виртуальные функции и абстрактные базовые классы. 3.4. Множественное наследование. 3.5. Указатели на компоненты класса. Доступ по указателю. 3.6. Объединения. 3.7. Ромбовидное наследование. Виртуальные базовые классы.	26	14	6	8	12	5	5	5	5
1	2	Раздел 4. Обобщенное программирование. Шаблоны функций и шаблоны классов. 4.1. Парадигма обобщенного программирования. C++ как язык поддерживающий обобщенное программирование. 4.2. Шаблоны функций. Оператор typename. 4.3. Шаблоны классов.	22	10	4	6	12	5	5	5	5
1	2	Раздел 5. Библиотека стандартных шаблонов (Standard Template Library, STL). 5.1. Итераторы STL. 5.2. Контейнеры STL. 5.3. Потоки STL. 5.4. Функциональные объекты STL. 5.5. Алгоритмы STL.	22	10	4	6	12	5	5	5	5
1	2	Раздел 6. Кроссплатформенная библиотека для создания многооконных приложений SDL 2.x. 6.1. Модель множества окон отображения библиотеках SDL 2.x. Понятия окна, визуализатора и текстуры в SDL 2.x. Связь поверхности отображения и текстуры отображения SDL 2.x. 6.2. Модель обработки событий в SDL 2.x. Работа с очередью событий SDL 2.x. 6.3. Обзор дополнительных библиотек семейства Simple DirectMedia Layer 2.x. 6.4. Организация ввода данных в графическом режиме с использованием SDL 2.x. 6.5. Способы программного построения двумерных движущихся изображений в SDL 2.x.	26	14	6	8	12	5	5	5	5
1	2	Раздел 7. Расширение возможностей языка C++. 7.1. Основные версии стандартов C++ и широко применяемые расширения стандартов. 7.2. Библиотеки C++ не входящие в стандарт.	13	4	4	0	9	5	5	5	5
Всего за 2 семестр			144	68	34	34	76	40	40	40	40
2	3	Раздел 8. Общезыковая среда выполнения (CLR). Единая система типов .NET Framework. 8.1. Общезыковая среда выполнения (CLR). Базовая библиотека классов (BCL). Понятие сборки (assembly). Метаданные и манифест сборки. 8.2 Единая система типов .NET Framework. Типы-значения (value types) и ссылочные типы (reference types). Класс System.Object как общий базовый класс всех типов C#. Упаковка(boxing) и распаковка(unboxing) типов значений. Анонимные типы. Допустимые преобразования типов. Тип dynamic. 8.3 Структура программы на языке C#. Пространства имен.	12	4	2	2	8	8	8	8	8
2	3	Раздел 9. Массивы, строки, работа с файлами. 9.1 Массивы. Определение и инициализация. Допустимые	14	6	2	4	8	6	6	6	6

		приведения типов массивов. 9.2 Классы System.String и System.Text.StringBuilder для работы со строками. Классы для работы с файлами и каталогами, потоковый класс System.IO.FileStream, классы System.IO.BinaryReader и System.IO.BinaryWriter. 9.3 Понятие потока в контексте работы с файлами.									
2	3	Раздел 10. Пространства имен. Классы, структуры, интерфейсы. 10.1 Классы и структуры в C#. Сравнение написание классов на языке C++ и C#. Частичные классы. Частичные методы. Свойства и индексы. Автореализуемые свойства. Модификаторы ref и out. Методы с переменным числом параметров. Модификатор params. Перегрузка операторов. 10.2 Операторы as и is. Тип интерфейса - определение и реализация. Явная и неявная реализация интерфейса. Реализация интерфейсов и наследование.	20	12	8	4	8	6	6	6	6
2	3	Раздел 11. Исключения. Типы с явным освобождением ресурсов. Сборщик мусора. 11.1 Механизм исключений. Блоки catch и finally. Иерархия библиотечных классов-исключений. 11.2 Жизненный цикл объекта. Деструкторы и метод Finalize. Сборщик мусора. Типы с явным освобождением ресурсов. Сравнение подходов к решению задачи освобождения ресурсов в языках C# и C++.	13	4	2	2	9	6	6	6	6
2	3	Раздел 12. Типы-коллекции и универсальные (обобщенные) коллекции. 12.1 Пространство имен System.Collections. Итераторы. Блок итератора. Оператор yield. Интерфейсы IEnumerable и IEnumerator. Оператор foreach. Интерфейсы ICollection и IList. Класс ArrayList. Интерфейс IDictionary и класс Hashtable. Интерфейсы IComparable и IComparer. 12.2 Интерфейсы ICollection, IList и IDictionary. Классы List и Dictionary.	20	12	8	4	8	6	6	6	6
2	3	Раздел 13. Делегаты и события. 13.1 Тип delegate. Классы System.Delegate и System.MulticastDelegate. Анонимные методы. Обобщенные делегаты. 13.2 Определение и реализация событий. Свойства события (event properties). События и интерфейсы. Делегаты EventHandler и EventHandler. Тип System.EventArgs. Интерфейс System.ComponentModel.INotifyPropertyChanged.	17	8	4	4	9	6	6	6	6
2	3	Раздел 14. Дополнительные главы языка C#. 14.1 Механизм сериализации. Виды сериализации Версия сборки. Сборки со строгим именем. Механизм отражения (reflection). Класс System.Type. Атрибуты. Определение пользовательских атрибутов. 14.2 Взаимодействие управляемого и неуправляемого кода. Сервис PInvoke. Атрибуты DllImport и MarshalAs. Маршалинг типов-значений и ссылочных типов. 14.3 Глобализация и локализация приложения. Региональные настройки (culture).	16	8	4	4	8	6	6	6	6
2	3	Раздел 15. Разработка графических приложений на языке C#. 15.1 Интерфейс GDI+. Пространства имен System.Drawing, System.Drawing.Drawing2D, System.Drawing.Imaging, и System.Drawing.Text. 15.2 Класс Graphics и его методы. Отрисовка графических примитивов.	15	6	2	4	9	8	8	8	8
2	3	Раздел 16. Разработка оконных приложений на языке C#. 16.1 Технология Windows Forms. Структура проекта. Ресурсы проекта. Архитектурный шаблон MVC. Особенности создания приложения.	17	8	2	6	9	8	8	8	8
Всего за 3 семестр			144	68	34	34	76	60	60	60	60
Всего по дисциплине			288	136	68	68	152	100	100	100	100

3.2. Аудиторный практикум

№ п/п	Номер и наименование раздела дисциплины	Тема практического занятия	Объем, ауд. часов
1	Раздел 2. Стандартные и пользовательские типы данных в C++.	Стандартные и пользовательские типы данных в C++. Классы в C++: определение и использование.	2
2	Обработка исключений. Инкапсуляция и статический полиморфизм в C++.	Выполнение индивидуальной практической работы 2 (ИПР-1): описание класса с выводом отладочной информации во всех методах класса, написание программы тестирования всех методов класса с интерфейсом пользователя на основе консольного текстового меню.	4
3	Раздел 3. Наследование и динамический полиморфизм в C++.	Наследование. Иерархия классов. Виртуальные функции. Динамический полиморфизм в C++. Виртуальные деструкторы. Оператор typeid.	2
4		Чисто виртуальные функции и абстрактные базовые классы. Множественное наследование. Ромбовидное наследование. Виртуальные базовые классы.	2
5		Выполнение индивидуальной практической работы 2 (ИПР-2): разработка иерархии классов из заданного базового и заданных производных классов, с указанными в задании обязательными и дополнительными компонентами, написание программы тестирования всех методов всех класса в иерархии, с	4

		интерфейсом пользователя на основе консольного текстового меню.	
6		Шаблоны функций и шаблоны классов в C++. Оператор <code>typename</code> .	2
7	Раздел 4. Обобщенное программирование. Шаблоны функций и шаблоны классов.	Выполнение индивидуальной практической работы 3 (ИПР-3): разработка шаблона заданной функции и написание программы ее тестирования для указанных типов данных, разработка шаблона класса абстрактного типа данных и написание программы с интерфейсом пользователя на основе консольного текстового меню для проверки всех методов классов, создаваемых на основе шаблона для указанных типов данных.	4
8	Раздел 5. Библиотека стандартных шаблонов (Standard Template Library, STL).	Обобщенное программирование с использованием библиотеки стандартных шаблонов (Standard Template Library, STL) C++	2
9		Выполнение индивидуальной практической работы 4 (ИПР-4): использование стандартного шаблона и написание программы тестирования для указанных типов данных и написание программы с интерфейсом пользователя на основе консольного текстового меню для проверки всех методов классов, создаваемых на основе шаблона для указанных типов данных.	4
10	Раздел 6.	Способы программного построения двумерных движущихся изображений.	2
11	Кроссплатформенная библиотека для создания многооконных приложений SDL 2.x.	Способы обеспечения синхронного и асинхронного выполнения частей программы средствами SDL 2.x. Таймеры SDL 2.x. Взаимосвязь механизма событий и таймеров в SDL 2.x.	2
12		Выполнение индивидуальной практической работы 5 (ИПР-5): написание программы построения графика функции и программы построения двумерного движущегося изображения	4
Всего за 2 семестр			34
13	Раздел 8. Общезыковая среда выполнения (CLR). Единая система типов .NET Framework.	Система типов C#, классы: определение и использование	2
14	Раздел 9. Массивы, строки, работа с файлами.	Работа с массивами, строками. Определение потоковых классов	2
15		Выполнение индивидуальной практической работы 6 (ИПР-6): работа со строками и файлами с использованием потоков	2
16	Раздел 10. Пространства имен. Классы, структуры, интерфейсы.	Сравнение написания классов в C++ и C#. Определение интерфейса. Использование интерфейса в механизме наследовани	2
17		Выполнение индивидуальной практической работы 7 (ИПР-7): разработка иерархии классов с использованием интерфейсов, абстрактных классов и других механизмов работы с наследованием	2
18	Раздел 11. Исключения. Типы с явным освобождением ресурсов. Сборщик мусора.	Работа с механизмом исключений. Жизненный цикл объекта. Работа со сборщиком мусора. Поколения жизни объекта	1
19		Выполнение индивидуальной практической работы 8 (ИПР-8): сравнение подходов к освобождению ресурсов в языках C++ и C#	1
20	Раздел 12. Типы-коллекции и универсальные (обобщенные) коллекции.	Понятие коллекции. Работа с коллекцией. Работа с итераторами. Универсальные интерфейсы коллекций.	4
21	Раздел 13. Делегаты и события.	Механизм делегатов. Реализация событий с использованием делегатов.	4
22	Раздел 14. Дополнительные главы языка C#.	Выполнение индивидуальной практической работы 9 (ИПР-9): изучение взаимодействия управляемого и неуправляемого кода	4
23	Раздел 15. Разработка	Отрисовка графических примитивов с использованием класса	4

	графических приложений на языке C#.	Graphics	
24	Раздел 16. Разработка оконных приложений на языке C#.	Способы создания оконных приложений с использованием технологии Windows Forms	6
Всего за 3 семестр			34

3.3. Самостоятельная работа студента (СРС)

№ п/п	Номер и наименование раздела дисциплины	Содержание учебного задания	Объем, часов
1	Раздел 1. Основные понятия объектно-ориентированного программирования (ООП).	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	5
2	Раздел 2. Стандартные и пользовательские типы данных в C++. Обработка исключений. Инкапсуляция и статический полиморфизм в C++.	Подготовка к практическим занятиям	4
3		Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	6
4		Оформление отчета по ИПР-1	4
5	Раздел 3. Наследование и динамический полиморфизм в C++.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	6
6		Подготовка к практическим занятиям	4
7		Оформление отчета по ИПР-2	2
8	Раздел 4. Обобщенное программирование. Шаблоны функций и шаблоны классов.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	8
9		Подготовка к практическим занятиям	2
10		Оформление отчета по ИПР-3	2
11	Раздел 5. Библиотека стандартных шаблонов (Standard Template Library, STL).	Оформление отчета по ИПР-4	2
12		Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	6
13		Подготовка к практическим занятиям	4
14	Раздел 6. Кроссплатформенная библиотека для создания многооконных приложений SDL 2.x.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	8
15		Подготовка к практическим занятиям	2
16		Оформление отчета по ИПР-5	2
17	Раздел 7. Расширение возможностей языка C++.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	9
Всего за 2 семестр			76
18	Раздел 8. Общезыковая среда выполнения (CLR). Единая система типов .NET Framework.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	7
19		Выполнение этапа 1 курсовой работы	1
20	Раздел 9. Массивы, строки, работа с файлами.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	4
21		Оформление отчета по ИПР-6	2
22		Выполнение этапа 2 курсовой	1

		работы	
23		Выполнение этапа 3 курсовой работы	1
24	Раздел 10. Пространства имен. Классы, структуры, интерфейсы.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	2
25		Выполнение этапа 5 курсовой работы	3
26		Оформление отчета по ИПР-7	1
27		Выполнение этапа 4 курсовой работы	2
28	Раздел 11. Исключения. Типы с явным освобождением ресурсов. Сборщик мусора.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	3
29		Оформление отчета по ИПР-8	1
30		Подготовка к практическим занятиям	1
31		Выполнение этапа 6 курсовой работы(часть 1)	4
32	Раздел 12. Типы-коллекции и универсальные (обобщенные) коллекции.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	2
33		Выполнение этапа 6 курсовой работы (часть 2)	6
34	Раздел 13. Делегаты и события.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	9
35	Раздел 14. Дополнительные главы языка C#.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	5
36		Оформление отчета по ИПР-9	3
37	Раздел 15. Разработка графических приложений на языке C#.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	9
38	Раздел 16. Разработка оконных приложений на языке C#.	Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	5
39		Оформление и подготовка к защите курсовой работы	4
Всего за 3 семестр			76

3.4. Курсовая работа

СОДЕРЖАНИЕ ЭТАПА	ПЕРИОД ИСПОЛНЕНИЯ (недели семестра)	ПЛАНИРУЕМОЕ ВРЕМЯ (час)
Этап 1. Анализ задачи, формулирование ограничений на конкретную реализацию. Описание предполагаемого процесса взаимодействия пользователя с программой.	1 - 2	1
Этап 2. Объектно-ориентированный анализ и проектирование: выявление классов, объектов и их отношений.	3 - 4	1
Этап 3. Выполнение программной реализации основных классов, написание демонстрационно-тестирующей программы.	5 - 6	1
Этап 4. Программная реализация каркаса графического интерфейса с использованием выбранного языка программирования и технологий	7 - 9	2
Этап 5. Программная реализация необходимого взаимодействия средствами выбранной технологии	10 - 11	3

Этап 6. Подготовка окончательного варианта программы, включая встроенные справочные материалы. Подготовка документации для пользователя	12 - 16	10
Всего за 3 семестр		18

4. ФОРМЫ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

СЕМЕСТР	НЕДЕЛИ СЕМЕСТРА																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2				Отч. по ПЗ		ДР		Отч. по ПЗ		ДР	Отч. по ПЗ			Отч. по ПЗ		ДР	Отч. по ПЗ, диф. зач.
3				Отч. по ПЗ		ДР		Отч. по ПЗ		ДР		Отч. по ПЗ				ДР	Отч. по ПЗ, КР, диф. зач.

Условные обозначения:

- ДР – диагностическая работа;
- Отч. по ПЗ – отчет по практическому заданию;
- КР – курсовая работа;
- диф. зач. – дифференцированный зачет;
- диф. зач. – дифференцированный зачет.

Текущий контроль успеваемости студентов проводится в дискретные временные интервалы в следующих формах:

- диагностическая работа;
- отчет по практическому заданию;
- курсовая работа.

Промежуточная аттестация проводится в формах:

- дифференцированный зачет.

5. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

5.1. Основная литература по дисциплине:

1. А. Н. Гуцин. . Применение библиотеки SDL для разработки программ на языке С. СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014, 50 экз.
2. А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#. СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022, 290 экз.
3. В. И. Юров. . Assembler. М.: Питер, 2006, 59 экз.
4. В. И. Юров. . Assembler. СПб.: Питер, 2010, эл. рес.
5. Дж. Бишоп, Н. Хорспул. . С# в кратком изложении. М.: БИНОМ. Лаборатория знаний, 2005, 5 экз.
6. И. Г. Головин, И. А. Волкова. . Языки и методы программирования. М.: Академия, 2016, 50 экз.
7. Н. А. Тюкачёв, В. Г. Хлебостроев. . С#. Программирование 2D и 3D векторной графики. Санкт-Петербург: Лань, 2022, эл. рес.
8. О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня. СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014, эл. рес.
9. О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня. СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014, 60 экз.
10. С. С. Сосинская. . Использование языка С# в различных информационных технологиях. Старый Оскол: ТНТ, 2020, эл. рес.
11. Т. А. Павловская. . С/С++. Программирование на языке высокого уровня . Санкт-Петербург: Питер, 2021, эл. рес.
12. Э. Троелсен. . С# и платформа .NET. СПб.: Питер, 2005, 20 экз.
13. Ю. А. Щупак. . Win32 API. Разработка приложений для Windows. СПб.: Питер, 2008, 48 экз.

5.2. Дополнительная литература по дисциплине:

не требуется.

5.3. Периодические издания:

не требуются.

5.4. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины, электронные библиотечные системы:

1. <https://urait.ru/> — Образовательная платформа «Юрайт». Для вузов и ссузов.;;
2. <https://e.lanbook.com/> — ЭБС Лань;;
3. <http://sourceforge.net/projects/orwelldevcpp/> — Dev-C++ download | SourceForge.net;;
4. <http://www.codeblocks.org/> — Code::Blocks - Code::Blocks;;
5. <http://scholar.google.ru/> — Академия Google;;
6. <http://www.open-std.org/jtc1/sc22/wg21/> — ISO/IEC JTC1/SC22/WG21 - The C++ Standards Committee - ISO C++;;
7. <http://library.voenmeh.ru/jirbis2/> ; — Фундаментальная библиотека БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова;
8. <https://docs.microsoft.com/> — Developer tools, technical documentation and coding examples | Microsoft Docs;;

Современные профессиональные базы данных:

1. <https://rusneb.ru> – Национальная электронная библиотека (НЭБ);
2. <https://cyberleninka.ru/> - Научная электронная библиотека «Киберленинка»;
<http://www.rfbr.ru/rffi/ru/library> - Полнотекстовая электронная библиотека Российского фонда фундаментальных исследований.

Информационные справочные системы:

1. Техэксперт – Информационный портал технического регулирования: Нормы, правила, стандарты РФ;
2. http://library.voenmeh.ru/jirbis2/index.php?option=com_irbis&view=irbis&Itemid=457 - БД ГОСТов собственной генерации БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова;
3. <http://www.consultant.ru/>- КонсультантПлюс- информационный портал правовой информации.

5.5. Программное обеспечение:

1. Code::Blocks;
2. Linux;
3. Microsoft Visual Studio Community.

5.6. Информационные технологии:

взаимодействие с обучающимися посредством ЭИОС Moodle БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова.

6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

6.1. Лекционные занятия:

специализированные требования по оборудованию отсутствуют; аудитория с посадочными местами по количеству студентов; доска.

6.2. Практические занятия:

1. Проектор;
2. Code::Blocks;
3. Linux;
4. Microsoft Visual Studio Community.

6.3. Прочее:

1. рабочее место преподавателя, оснащенное компьютером с доступом в Интернет;
2. рабочие места студентов, оснащенные компьютерами с доступом в Интернет, предназначенные для работы в электронной образовательной среде.

Аннотация рабочей программы

Дисциплина **ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ПРОГРАММИРОВАНИЕ** является дисциплиной **обязательной части блока 1** программы подготовки по направлению *09.03.01 Информатика и вычислительная техника*. Дисциплина реализуется на факультете *О Естественнотехнический БГТУ "ВОЕНМЕХ"* им. Д.Ф. Устинова кафедрой *О7 Информационные системы и программная инженерия*.

Дисциплина нацелена на формирование *компетенций*:

ПСК-1.1 способность разрабатывать требования и проектировать программное обеспечение;

ОПК-2 способность понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности;

ОПК-8 способность разрабатывать алгоритмы и программы, пригодные для практического применения;

ОПК-9 способность осваивать методики использования программных средств для решения практических задач.

Содержание дисциплины охватывает круг вопросов, связанных с использованием языков программирования высокого уровня при разработки программного обеспечения. Основное внимание уделяется парадигмам объектно-ориентированного программирования и обобщенного программирования, а также расширения языков программирования высокого уровня сторонними библиотеками, рассматриваются вопросы разработки программ с графическим пользовательским интерфейсом и взаимодействия программ с программными интерфейсами операционных систем. Также приводятся общие сведения о процессах, потоках, синхронном и асинхронном взаимодействии программ и их частей. В качестве основы для практических примеров рассматриваются язык программирования C++, библиотека стандартных шаблонов C++ STL, библиотека 2.x (SDL 2.x). Так же в качестве основы для практических примеров рассматриваются язык программирования C# и основанные на нем интерфейсы программирования приложений Windows Forms и WPF, а также внутренние библиотеки программной платформы .NET Core.

Программой дисциплины предусмотрены следующие **виды контроля**:

Текущий контроль успеваемости студентов проводится в дискретные временные интервалы в следующих формах:

- диагностическая работа;
- отчет по практическому заданию;
- курсовая работа.

Промежуточная аттестация проводится в формах:

- дифференцированный зачет.

Общая трудоемкость освоения дисциплины составляет **8 з.е., 288 ч.** Программой дисциплины предусмотрены лекционные занятия (**68 ч.**), практические занятия (**68 ч.**), самостоятельная работа студента (**152 ч.**).

ТЕХНОЛОГИИ И ФОРМЫ ОБУЧЕНИЯ

Рекомендации по освоению дисциплины для студента

Трудоемкость освоения дисциплины составляет 288 ч., из них 136 ч. аудиторных занятий, и 152 ч., отведенных на самостоятельную работу студента.

Рекомендации по распределению учебного времени по видам самостоятельной работы и разделам дисциплины приведены в таблице.

Контроль освоения дисциплины производится в соответствии с Положением о текущем, рубежном контроле успеваемости и промежуточной аттестации обучающихся.

Формы контроля и критерии оценивания приведены в приложении 3 к Рабочей программе.

Наименование работы	Рекомендуемая литература	Трудоемкость, час.
Раздел 1. Основные понятия объектно-ориентированного программирования (ООП).		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1) Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (1)	5
Итого по разделу 1		5
Раздел 2. Стандартные и пользовательские типы данных в С++. Обработка исключений. Инкапсуляция и статический полиморфизм в С++.		
Подготовка к практическим занятиям	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (2)	4
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (2) Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (2) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (2)	6
Оформление отчета по ИПР-1	СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (2)	4
Итого по разделу 2		14
Раздел 3. Наследование и динамический полиморфизм в С++.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (1-3) В. И. Юров. . Assembler: СПб.: Питер, 2010 (3) В. И. Юров. . Assembler: М.: Питер, 2006 (3) Ю. А. Щупак. . Win32 API. Разработка приложений для Windows: СПб.: Питер, 2008 (5)	6
Подготовка к практическим занятиям		4
Оформление отчета по ИПР-2		2
Итого по разделу 3		12

Раздел 4. Обобщенное программирование. Шаблоны функций и шаблоны классов.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (1-3) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-3)	8
Подготовка к практическим занятиям	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-3)	2
Оформление отчета по ИПР-3	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1-3)	2
Итого по разделу 4		12
Раздел 5. Библиотека стандартных шаблонов (Standard Template Library, STL).		
Оформление отчета по ИПР-4	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (5)	2
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1-3) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (5)	6
Подготовка к практическим занятиям	Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (5)	4
Итого по разделу 5		12
Раздел 6. Кроссплатформенная библиотека для создания многооконных приложений SDL 2.x.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (1-4) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1-3) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-4)	8
Подготовка к практическим занятиям	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-4)	2
Оформление отчета по ИПР-5	О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-4)	2
Итого по разделу 6		12
Раздел 7. Расширение возможностей языка С++.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	А. Н. Гуцин. . Применение библиотеки SDL для разработки программ на языке С: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (1-4) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (5) О. В. Арипова, А. Н. Гуцин, О. А. Палехова. . Программирование на языке высокого уровня: СПб.БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2014 (5) Т. А. Павловская. . С/С++. Программирование на языке высокого уровня : Санкт-Петербург: Питер, 2021 (5)	9
Итого по разделу 7		9

Раздел 8. Общеязыковая среда выполнения (CLR). Единая система типов .NET Framework.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (1-2) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1) С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (1)	7
Выполнение этапа 1 курсовой работы	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (1-2)	1
Итого по разделу 8		8
Раздел 9. Массивы, строки, работа с файлами.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (3) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (3)	4
Оформление отчета по ИПР-6	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (3)	2
Выполнение этапа 2 курсовой работы	С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (1-2)	1
Выполнение этапа 3 курсовой работы		1
Итого по разделу 9		8
Раздел 10. Пространства имен. Классы, структуры, интерфейсы.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (3) Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (5)	2
Выполнение этапа 5 курсовой работы	А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#: СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022 (1)	3
Оформление отчета по ИПР-7	С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (3)	1
Выполнение этапа 4 курсовой работы		2
Итого по разделу 10		8
Раздел 11. Исключения. Типы с явным освобождением ресурсов. Сборщик мусора.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (3) Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (5)	3
Оформление отчета по ИПР-8	С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (2)	1
Подготовка к практическим занятиям		1
Выполнение этапа 6 курсовой работы(часть 1)	И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (1)	4
Итого по разделу 11		9
Раздел 12. Типы-коллекции и универсальные (обобщенные) коллекции.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (4-5) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (7-8) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (4) А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#: СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022 (2)	2
Выполнение этапа 6 курсовой работы (часть 2)	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (8)	6

Итого по разделу 12		8
Раздел 13. Делегаты и события.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (10) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (3) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (5) С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (3)	9
Итого по разделу 13		9
Раздел 14. Дополнительные главы языка С#.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (20) С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (7) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (3) А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#: СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022 (3)	5
Оформление отчета по ИПР-9	Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (7)	3
Итого по разделу 14		8
Раздел 15. Разработка графических приложений на языке С#.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (17) С. С. Сосинская. . Использование языка С# в различных информационных технологиях: Старый Оскол: ТНТ, 2020 (5) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (5) А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#: СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022 (4) И. Г. Головин, И. А. Волкова . . Языки и методы программирования: М.: Академия, 2016 (10)	9
Итого по разделу 15		9
Раздел 16. Разработка оконных приложений на языке С#.		
Изучение предусмотренных программой дидактических единиц по рекомендуемой литературе	Э. Троелсен. . С# и платформа .NET: СПб.: Питер, 2005 (18) А.А. Бармина, К. В. Вальштейн. . Программирование на языке С#: СПб.: Изд-во БГТУ "ВОЕНМЕХ" им. Д. Ф. Устинова, 2022 (4) Дж. Бишоп, Н. Хорспул. . С# в кратком изложении: М.: БИНОМ. Лаборатория знаний, 2005 (1-5)	5
Оформление и подготовка к защите курсовой работы	Н. А. Тюкачёв, В. Г. Хлебостроев. . С#. Программирование 2D и 3D векторной графики: Санкт-Петербург: Лань, 2022 (1-3)	4
Итого по разделу 16		9

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Фонд оценочных средств, позволяющие оценить результаты обучения по данной дисциплине, включают в себя:

- диагностическая работа
- отчет по практическому заданию;
- курсовая работа;
- дифференцированный зачет;
- дифференцированный зачет.

Критерии оценивания

Диагностическая работа

Диагностическая работа проводится в форме теста в ЭИОС Moodle:

- при правильном ответе менее чем на 60% вопросов - не аттестация;
- при правильном ответе на 60% вопросов и более - аттестация.

Отчет по практическому заданию

Во втором семестре необходимо выполнить пять ИПР. В третьем - четыре ИПР.

По всем ИПР необходимо успешное выполнение требования общей и вариативной части задания, включая предъявление в работе самостоятельно написанных соответствующих программ, в том числе на основе указанных примеров, если это предусмотрено заданием.

Отчет по ИПР:

Дополнительно к представлению всех результатов выполнения ИПР в электронной форме предусмотрено оформление отчетов, состоящих из титульного листа, вариативной части задания и основных результатов работы программы, а также наиболее соответствующих теме задания фрагментов разработанных программ.

Защита ИПР:

Защита ИПР предусматривает обсуждение порядка решения предусмотренных его тематикой задач, включая проверку усвоения студентом соответствующих сведений из теории и степени самостоятельности при написании предъявляемых программ.

Подробные критерии оценивания ИПР указаны в технологической карте дисциплины, размещённой в курсе в ЭИОС.

Курсовая работа

Выполненные курсовые работы представляются в электронной форме в виде подготовленных к сборке исходных текстов и полностью готовой к выполнению программы для тестирования преподавателем и электронной версии пояснительной записки, оформленной в соответствии с Положением по содержанию, оформлению, организации выполнения и защиты курсовых проектов и курсовых работ. При успешном тестировании программы и проверке соответствия пояснительной записки требованиям Положения и требованиям задания на данную курсовую работу, дается разрешение на ее печать без исходных текстов программ (они заменяются на «приложение в электронной форме»). При наличии распечатанной пояснительной записки студент допускается к защите КРР.

Критерии оценивания:

Курсовая работа допускается к защите при следующих условиях:

- предъявляемая программа работоспособна;
- программа выполнена соответствии с заданием;
- электронная и печатная версии пояснительной записки соответствуют установленным требованиям.

Оценка написанной КР:

- Работа выполнена, но не соответствует теме либо не использованы требуемые технологии, либо не реализованы все заявленные требования – 3 балла
- Работа выполнена в соответствии с темой, реализовано 85% заявленных возможностей, пользовательский интерфейс не предусматривает проверки ввода и не исправлены технические ошибки

(утечки памяти, программа некорректно завершает работу) - 6 баллов

- Работа выполнена, реализованы все заявленные возможности, пользовательский интерфейс содержит фразы на английском языке, отсутствует инструкция для пользователя по работе с программой - 9 баллов

- Работа выполнена, реализованы все возможности, ошибок в работе и интерфейсе не выявлено - 10 баллов

Оценка содержания пояснительной записки к курсовой работе:

- Пояснительная записка не содержит описания структуры разработанной программы, тестирование программы не произведено, продемонстрирован исключительно пользовательский интерфейс – 2 балла

- Пояснительная записка содержит описание структуры разработанной программы без использования диаграмм классов, тестирование программы не произведено, продемонстрирован пользовательский интерфейс и результаты работы программы – 3 балла

- Структура программы описана минимум одной диаграммой классов, описан базовый процесс тестирования, записка имеет четкую структуру в виде выделенных разделов и подразделов – 4 балла

- Структура программы описана диаграммами нескольких типов, полностью описан процесс тестирования, записка имеет четкую структуру в виде выделенных разделов и подразделов - 5 баллов

Оценка оформления, стиля пояснительной записки

- Пояснительная записка оформлена с нарушениями, язык работы не соответствует научному стилю, некорректно оформленные заимствования, некорректно оформлен список источников – 2 балла

- Пояснительная записка оформлена с нарушениями, язык работы не соответствует научному стилю, есть замечания к оформлению списка источников – 3 балла

- Есть отдельные замечания к оформлению и стилю изложения, оформлению списка источников – 4 балла

- Нет замечаний к оформлению и стилю изложения, оформлению списка источников – 5 баллов

Максимальное количество баллов – 20

Оценка «отлично» - 17-20 баллов

Оценка «хорошо» - 13-16 баллов

Оценка «удовлетворительно» - 10-12 баллов

Оценка «не защитил» - меньше 10 или работа не была предъявлена

Дифференцированный зачет

Дифференцированный зачет проводится в виде электронного тестирования в ЭИОС.

В тесте 30 вопросов с максимальным баллом 30.

Шкала оценивания:

0 - 15 баллов - неудовлетворительно.

16 - 20 баллов - удовлетворительно.

21 - 25 баллов - хорошо.

26 - 30 баллов - отлично.

На тест дается 90 минут. Можно улучшить свой результат каждые 3 суток.

Всего попыток - 3.

При выполнении и защите всех ПЗ до начала промежуточной аттестации предусмотрено повышение оценки на одну ступень начиная с оценки "зачтено-удовлетворительно".

При выполнении и защите всех ПЗ в усложненном варианте предусмотрена оценка "зачтено-отлично" по результатам работы в семестре.

Также предусмотрено получение оценки согласно набранным во время семестра баллам, согласно размещенной в курсе в ЭОИС технологической карте.

Дифференцированный зачет

Дифференцированный зачет проводится в виде электронного тестирования в ЭИОС.

В тесте 30 вопросов с максимальным баллом 30.

Шкала оценивания:

0 - 15 баллов - не зачтено.

16 - 20 баллов - удовлетворительно.

21 - 25 баллов - хорошо.

26 - 30 баллов - отлично.

На тест дается 90 минут. Можно улучшить свой результат каждые 3 суток.

Всего попыток - 3.

При выполнении и защите всех ПЗ до начала промежуточной аттестации предусмотрено повышение оценки на одну ступень начиная с оценки "зачтено-удовлетворительно".

При выполнении и защите всех ПЗ в усложненном варианте предусмотрена оценка "зачтено-отлично" по результатам работы в семестре.

Также предусмотрено получение оценки согласно набранным во время семестра баллам, согласно размещённой в курсе в ЭОИС технологической карте.

КУРС	СЕМЕСТР	Наименование разделов и дидактических единиц	ВСЕГО	Аудиторные занятия в контактной форме			Самостоятельная работа студентов	Формируемая компетенция, %				НАИМЕНОВАНИЕ ОЦЕНОЧНОГО СРЕДСТВА
				ВСЕГО	Лекции	Практические занятия		ПСК-1.1	ОПК-2	ОПК-8	ОПК-9	
1	2	Раздел 1. Основные понятия объектно-ориентированного программирования (ООП).	9	4	4	0	5	10	10	10	10	Отчет по практическому заданию
1	2	Раздел 2. Стандартные и пользовательские типы данных в C++. Обработка исключений. Инкапсуляция и статический полиморфизм в C++.	26	12	6	6	14	5	5	5	5	Отчет по практическому заданию
1	2	Раздел 3. Наследование и динамический полиморфизм в C++.	26	14	6	8	12	5	5	5	5	Отчет по практическому заданию
1	2	Раздел 4. Обобщенное программирование. Шаблоны функций и шаблоны классов.	22	10	4	6	12	5	5	5	5	Отчет по практическому заданию
1	2	Раздел 5. Библиотека стандартных шаблонов (Standard Template Library, STL).	22	10	4	6	12	5	5	5	5	Отчет по практическому заданию
1	2	Раздел 6. Кроссплатформенная библиотека для создания многооконных приложений SDL 2.x.	26	14	6	8	12	5	5	5	5	Отчет по практическому заданию
1	2	Раздел 7. Расширение возможностей языка C++.	13	4	4	0	9	5	5	5	5	Отчет по практическому заданию
Всего за 2 семестр			144	68	34	34	76	40	40	40	40	
2	3	Раздел 8. Общеязыковая среда выполнения (CLR). Единая система типов .NET Framework.	12	4	2	2	8	8	8	8	8	Отчет по практическому заданию
2	3	Раздел 9. Массивы, строки, работа с файлами.	14	6	2	4	8	6	6	6	6	Отчет по практическому заданию

2	3	Раздел 10. Пространства имен. Классы, структуры, интерфейсы.	20	12	8	4	8	6	6	6	6	Отчет по практическому заданию
2	3	Раздел 11. Исключения. Типы с явным освобождением ресурсов. Сборщик мусора.	13	4	2	2	9	6	6	6	6	Отчет по практическому заданию
2	3	Раздел 12. Типы- коллекции и универсальные (обобщенные) коллекции.	20	12	8	4	8	6	6	6	6	Отчет по практическому заданию
2	3	Раздел 13. Делегаты и события.	17	8	4	4	9	6	6	6	6	Отчет по практическому заданию
2	3	Раздел 14. Дополнительные главы языка C#.	16	8	4	4	8	6	6	6	6	Отчет по практическому заданию
2	3	Раздел 15. Разработка графических приложений на языке C#.	15	6	2	4	9	8	8	8	8	Отчет по практическому заданию
2	3	Раздел 16. Разработка оконных приложений на языке C#.	17	8	2	6	9	8	8	8	8	Курсовая работа
Всего за 3 семестр			144	68	34	34	76	60	60	60	60	
Всего по дисциплине			288	136	68	68	152	100	100	100	100	

Критерии оценивания

ПСК-1.1

- Вопросы открытого типа:
- № 1 Какой принцип ООП необходимо использовать, чтобы заменить конструкции if-else в данном фрагменте кода:

```
if (animal.IsCat()) { /* код */ }
```

```
else if (animal.IsDog()) { /* код */ }
```

```
else if (animal.IsKoala()) { /* код */ }
```

```
...
```

```
else if (animal.isMouse()) { /* код */ }
```

- № 2 Полиморфизм – это
- № 3 Дан текст программы

```
#include
```

```
using namespace std;
```

```
class Object
```

```
{
```

```
public:
```

```
Object() { cout << "Object::ctor()" << endl; }
```

```
~Object() { cout << "Object::dtor()" << endl; }
```

```
};
```

```
class Base
```

```
{
```

```
public:
```

```
Base() { cout << "Base::ctor()" << endl; }
```

```
virtual ~Base() { cout << "Base::dtor()" << endl; }
```

```
virtual void print() = 0;
```

```
};
```

```
class Derived: public Base
```

```
{
```

```
public:
```

```
Derived() { cout << "Derived::ctor()" << endl; }
```

```
~Derived() { cout << "Derived::dtor()" << endl; }
```

```
void print() {}
```

```
Object obj;
```

```
};
```

```
int main ()
```

```
{
```

```
    Base * p = new Derived;
```

```
    delete p;
```

```
    return 0;
```

```
}
```

В каком порядке будут вызываться деструкторы?

№ 4 Дан текст программы

```
#include
```

```
using namespace std;
```

```
class Object
```

```
{
```

```
public:
```

```
    Object() { cout << "Object::ctor()" << endl; }
```

```

~Object() { cout << "Object::~dtor()" << endl; }

};

class Base

{

public:

    Base() { cout << "Base::ctor()" << endl; }

    virtual ~Base() { cout << "Base::~dtor()" << endl; }

    virtual void print() = 0;

};

class Derived: public Base

{

public:

    Derived() { cout << "Derived::ctor()" << endl; }

    ~Derived() { cout << "Derived::~dtor()" << endl; }

    void print() {}

```

```
Object obj;
```

```
};
```

```
int main ()
```

```
{
```

```
Base * p = new Derived;
```

```
delete p;
```

```
return 0;
```

```
}
```

В каком порядке будут вызываться конструкторы?

№ 5 Имеется фрагмент программы:

```
#include
```

```
using namespace std;
```

```
class A
```

```
{
```

```

public:

    void f1(){cout << "A::f1 ";}

    virtual void f2(){cout << "A::f2 ";}

    virtual ~A(){}

};

class B: public A

{

public:

    void f1(){cout << "B::f1 ";}

    void f2(){cout << "B::f2 ";}

    ~B(){}

};

int main(void)

{

    A *ab = new B;

    B *bb = new B;

```

```
ab->f1();
```

```
bb->f1();
```

....Что будет выведено на экран в результате выполнения этого фрагмента?

№ 6 Шаблон функции - это

№ 7 Шаблон функции определен следующим образом:

```
template < class T >
```

```
int compare (T a, T b)
```

```
{
```

```
    return a > b ? 1 : a < b ? -1 : 0;
```

```
}
```

Чему будет равно значение выражения

```
compare ("1+1", "2")
```

№ 8 Как обозначается модель синтеза цвета светящегося пикселя на основе трех монохромных субпикселей красного, синего и зеленого цвета?

№ 9 Имеется фрагмент программы:

```
#include
```

```
using namespace std;
```

```
class A
```

```
{
```

```
public:
```

```
void f1(){cout << "A::f1 ";
```

```
virtual void f2(){cout << "A::f2 ";
```

```
virtual ~A(){}  
};
```

```
class B: public A
```

```
{
```

```
public:
```

```
void f1(){cout << "B::f1 ";
```

```
void f2(){cout << "B::f2 ";
```

```
~B(){}  
};
```



```
int main(void)
```

```
{
```

```
    A *ab = new B;
```

```
    B *bb = new B;
```

```
    ab->f2();
```

```
    bb->f2();
```

....Что будет выведено на экран в результате выполнения этого фрагмента?

№ 10

Базовый класс двунаправленного потока в C++

№ 11

Как исправить следующий код C#, чтобы он отработывал корректно? (пока что он в безопасном контексте и выдает ошибку CS0214)

```
UInt32 SumArray (byte *data, UInt32 len)
```

```
{
```

```
    UInt32 sum = 0;
```

```
    for(UInt32 i=0; i
```

```
        sum += *data++;
```

```
    return CRC8;
```

```
}
```

№ 12

Какое соглашение о вызове используют библиотеки семейства Win32API?

№ 13

Пусть Developer - определённый в C# с атрибутом [AttributeUsage(AttributeTargets.All)] пользовательский атрибут. Сработает ли следующая конструкция, и если нет, то почему?

```
static string dev = "Joan Smith";
```

```
[Developer(dev, "1", Reviewed = Верно)]
```

```
static bool myFun()
```

```
{    return Reviewed;    }
```

- № 14 Что такое атрибут в C#?
- № 15 Как узнать значение атрибута внутри сборки в C#?
- № 16 Что включает в себя полное имя сборки в C#?
- № 17 Как расшифровывается GAC?
- № 18 Какие форматы сериализации доступны в C#?
- № 19 Какой тип составляет ядро подсистемы рефлексии в C#, поскольку он инкапсулирует тип данных?
- № 20 Как расшифровывается TAP в контексте асинхронности в C#?
- Вопросы закрытого типа:*
- № 1 Каков общий формат определения шаблона функции в C++?

template <параметры шаблона> тип_результата имя_функции (параметры функции)

{ тело функции }

<template параметры_шаблона> тип_результата имя_функции (параметры функции)

{ тело функции }

тип_результата <параметры шаблона> template имя_функции (параметры функции)

{ тело функции }

template (параметры шаблона) тип_результата имя_функции (параметры функции)

{ тело функции }

- № 2 Какой способ необходимо использовать, чтобы объявить некоторый класс A дружественным некоторому классу B в C++?

Указать в определении класса A, что он друг классу B оператором friend class B;

Указать в определении каждого конструктора класса A, что он друг классу B оператором friend class B;

Указать в определении класса B, что у него есть друг класс A оператором friend class A;

Указать в определении любого из конструкторов класса B, что у него есть друг класс A оператором friend class A;

- № 3 Сколько дружественных классов можно объявлять в теле класса в C++?

Столько же, сколько всего членов класса

Сколько угодно

Столько, сколько классов считают данный класс дружественным себе

Не менее числа ранее объявленных классов

№ 4 В какой части тела класса (определения класса) могут располагаться объявления дружественных классов в C++?

Только в секции private:

Только в секции public:

Только в секциях private: или public:, но все объявления дружественных классов должны находиться только в одной секции

В любой

№ 5 В каких скобках задаются параметры шаблона функции в C++?

В угловых <>

В круглых ()

В квадратных []

В фигурных {}

№ 6 В каких скобках задаются параметры шаблонной функции в C++?

В угловых <>

В круглых ()

В квадратных []

В фигурных {}

№ 7 Имеется фрагмент программы

#include

```
using namespace std;
```

```
class A
```

```
{
```

```
public:
```

```
void f1(){cout << "A::f1" <
```

```
virtual void f2(){cout << "A::f2" <
```

```
virtual ~A(){}  
  
};
```

```
class B: public A
```

```
{
```

```
public:
```

```
void f1(){cout << "B::f1" <
```

```
void f2(){cout << "B::f2" <
```

```
~B(){}  
  
};
```

```
int main(void)
```

```
{
```

```
    A *aa = new A;
```

```
    A *ab = new B;
```

```
    B *ba = new A;
```

```
    B *bb = new B;
```

```
    ...
```

```
}
```

Какой или какие варианты создания объектов невозможны?

B *ba = new A

Ошибок нет, все варианты создания объектов допустимы

A *ab = new B; и B *ba = new A;

Ошибочны все, кроме B *bb = new B;

№ 8 Функтор это

класс, в котором определена операторная функция operator()

функциональный объект STL

особый метод класса

второе название конструктора

№ 9 Что означает "класс A является дружественным классу B"?

Все функции-члены класса А имеют доступ ко всем переменным-членам класса В, включая приватные

Все функции-члены класса А имеют доступ ко всем функциям-членам класса В, включая приватные

Все функции-члены класса А имеют доступ ко всем членам класса В, включая приватные

Все переменные-члены класса А имеют доступ ко всем членам класса В, включая приватные

№ 10 Что такое шаблон функции?

Определение функции, в которой типу обрабатываемых данных присвоено условное обозначение

Прототип функции, в котором вместо имен параметров указан условный тип

Определение функции, в котором указаны возможные варианты типов обрабатываемых параметров

Определение функции, в котором в прототипе указан условный тип, а в определении указаны варианты типов обрабатываемых параметров

№ 11 Компилятор "разворачивает" делегат в класс. Выберите методы, которые обязательно присутствуют в этом классе в C#.

Invoke

BeginInvoke

EndInvoke

Конструктор класса

№ 12 Сопоставьте следующие директивы в C# и их описание

#if

#ifdef

#define

#region

#using

Проверка определения идентификатора

Отсутствует в языке C#

	<p>Определяет конкретный идентификатор</p> <p>Указывает блок программы для возможности скрыть его</p> <p>Указывает независимый блок программы для переноса его в другой файл</p> <p>Определяет значение конкретного идентификатора</p> <p>Определяет настройки культуры региона</p> <p>Проверяет результат указанного выражения</p> <p>Отсутствует в языке C#</p> <p>Позволяет подключить классы из другой сборки</p>
№ 13	<p>Дан делегат <code>delegate T Factory();</code></p> <p>Этот делегат является ковариантным или контрвариантным в C#?</p> <p>ковариантный</p> <p>контрвариантный</p> <p>Невозможно определить</p> <p>и ковариантным, и контрвариантным</p> <p>таких свойств у делегата нет</p>
№ 14	<p>Что из перечисленного возможно сделать при помощи директивы <code>#define</code> в языке C#?</p> <p>Задать значение идентификатору</p> <p>Определить идентификатор</p> <p>Определить макрос</p> <p>Определить регион программы</p>
№ 15	<p>... в контексте .NET представляет особые типы, для представления в памяти используют одинаковый принцип выделения памяти как в управляемом, так и неуправляемых окружениях.</p> <p>Выберите верный термин</p> <p>Blittable</p> <p>Non-blittable</p> <p>Маршаллер</p> <p>Нативные</p>

- № 16 Соотнесите типы из WinAPI с типами, которые эквивалентны им в C# при маршаллинге
- BOOL
 LPSTR
 LPDWORD
 WORD
- bool
 sbyte*
 uint*
 char*
 unsigned int*
 ushort
- № 17 Выберите те реализации указателя на int которые потребуют использования ключевого слова unsafe в C#
- int *
 IntPtr
 fixed int
 int[]
- № 18 При создании собственного атрибута в C# необходимо описать класс, унаследованный от `Attribute`, использовать для описанного класса атрибут `AttributeUsage` и `AttributeTargets`
- Сопоставьте пропускам верные выражения из числа указанных ниже:
- Attribute
 AttributeUsage
 при необходимости определить свойства и конструкторы
 определить свойства и конструкторы
 реализовать в нём методы интерфейса `IAtribute`
 реализовать в нём методы `GetAttribute()` и `GetAttributeValue()`
 UserAttribute

CustomAttribute

Attributes

AttributeTarget

№ 19 Что из перечисленного может являться целью атрибута?

Класс

Поле

Конструктор

Сборка

Класс атрибута

№ 20 Выберите действия, необходимые для сериализации объекта.

Создание или получение потока

Создание форматтера

Вызов метода для сериализации

Закрытие потока

Закрытие форматтера

ОПК-2

Вопросы открытого типа:

№ 1 Все ли операторы языка C++ могут быть перегружены? Поясните ответ.

№ 2 Укажите зарезервированное ключевое слово для высвобождения памяти, выделенной оператором new в C++

№ 3 Почему при перегрузке инжектора в поток (операция <<) в C++ необходимо использовать дружественную функцию?

№ 4 Как изменить приоритет оператора при его перегрузке в C++?

№ 5 Значение какого типа возвращает конструктор в C++?

№ 6 Какими способами можно обратиться к компонентам объекта некоторого класса в C++?

№ 7 В каких случаях необходимо перегружать оператор присваивания в C++?

№ 8 Какой конструктор вызывается при создании каждого объекта в массиве, создаваемом оператором new[] в C++?

№ 9 Какой конструктор может быть вызван при создании объекта оператором new в C++ ?

№ 10 Дана программа на языке C++:

```
class C1 {  
    int data;  
  
public:  
    C1(int d=0){data=d;};  
    int getData(){return data;}  
};  
  
C1 f(C1 beta) {
```

```

C1 gamma;

gamma = beta;

return gamma;

}

int main(int n, char *c[]){

    C1 alpha;

    return f(alpha).getData();

}

```

Сколько каких конструкторов будет вызвано за время её выполнения?

- № 11 Какой уровень доступа к полям и методам типа struct установлен по умолчанию в C#?
- № 12 Какой уровень доступа к struct установлен по умолчанию в C#?
- № 13 Как задать метод с переменным числом параметров в C#?
- № 14 Какое округление используется при вызове Convert.ToInt32(4.5) и Convert.ToInt32(5.5) в C#?
- № 15 Какой формат вывода в C# необходим для вывода вещественного числа в шестнадцатеричном виде?
- № 16 Что выведет следующая строка в C#?

Console.WriteLine('m' + 'e' + 'o' + 'w');
- № 17 Как расшифровывается CIL?
- № 18 Как называется библиотека классов фреймворка .NET?
- № 19 Отличие в хранении value type и reference type в C#:
- № 20 Назовите отличие string от System.String в C#
Вопросы закрытого типа:
- № 1 Для доступа к элементам объекта в C++ используются

при обращении через имя объекта – точка, при обращении через указатель – операция «->»

при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «точка»

при обращении через имя объекта – точка, при обращении через указатель – два двоеточия

при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «->»

- № 2 Выберите верное утверждение относительно языка C++

И конструкторов, и деструкторов может быть много, при этом количество конструкторов должно быть не меньше количества деструкторов.

Конструкторов может быть много, а деструктор - только один.

В классе может быть один конструктор и несколько деструкторов.

В классе может быть только один конструктор и один деструктор.

№ 3 Какое минимальное количество конструкторов и деструкторов может быть в классе в C++?

2 конструктора, 1 деструктор

1 конструктор, 1 деструктор

3 конструктора, 1 деструктор

2 конструктора, 2 деструктора

№ 4 Какие методы создаются автоматически при объявлении класса в C++? Выберите наиболее полный вариант из предложенных

Конструктор по умолчанию, конструктор копирования, деструктор, оператор присваивания

Конструктор по умолчанию, конструктор копирования, оператор присваивания

Конструктор по умолчанию, конструктор копирования, деструктор

Конструктор по умолчанию, конструктор с параметрами, конструктор копирования, деструктор по умолчанию, деструктор с параметрами

№ 5 Пусть имеется некоторый класс A и объект a:

```
class A {  
    double Num;  
  
public:  
    double getNum();  
    void setNum(double d);  
} a;
```

Где-то в программе есть строка:

```
a.setNum(1.57);
```

Что происходит в данной строке в C++?

Вызов метода setNum() для объекта a непосредственно

Вызов метода setNum() для объекта a по указателю

Вызов метода setNum() для объекта a по ссылке

Вызов метода setNum() для объекта a через точку

№ 6 Какого спецификатора доступа в C++ нет?

public

internal

private

protected

№ 7 Операторами управления памятью в C++ являются

new, delete

malloc(), free()

new, delete, malloc(), free()

getMemory, freeMemory

№ 8 Пусть имеется некоторый класс A и определенные следующим образом переменные a и b:

```
class A {
```

```
    double Num;
```

```
public:
```

```
    double getNum();
```

```
    void setNum(double d);
```

```
} b, *a=&b;
```

Где-то в программе есть строка:

```
a->setNum(1.57);
```

Что происходит в данной строке в C++?

Вызов метода setNum класса A непосредственно для объекта a

Вызов метода setNum класса A непосредственно по указателю a

Вызов метода setNum класса A непосредственно по ссылке a

Вызов метода setNum класса A непосредственно для объекта b

№ 9 Переопределение операций в C++ имеет вид

имя_класса, ключевое слово operator, символ операции, в круглых скобках могут быть указаны аргументы

имя_класса, ключевое слово operation, символ операции

имя_класса, ключевое слово operator, список аргументов

имя_класса, два двоеточия, ключевое слово operator, символ операции

№ 10 Пусть A - имя класса, определенного пользователем. Тогда языковая конструкция A B; в C++ означает

создание объекта B класса A

объявление переменной A перечислимого типа B

инициализацию поля A объекта B

вызов метода B класса A

№ 11 Какое из следующих утверждений верно в C#?

Struct – это value type, Class – reference type

Struct – это reference type, Class –value type

И Struct и Class – это reference type

Тип (value или reference) зависит от типа полей

№ 12 Выберите допустимые типы конструкторов в C#

Конструктор по умолчанию

Конструктор копирования

Конструктор с параметрами

Статический конструктор

№ 13 Что из перечисленного нельзя объявлять с модификатором static в C#?

Конструктор

Свойство

Финализатор

- № 14 Каково поведение Equals (object objA, object objB) в C#?
- Если оба объекта не представляют одну и ту же ссылку на объект и ни один из них не имеет значения NULL, то вызывает objA.Equals (objB) и возвращает результат.
- Если оба объекта не представляют одну и ту же ссылку на объект и ни один из них не имеет значения NULL, то вызывает objB.Equals (objA) и возвращает результат.
- Является псевдонимом для objA.Equals (objB)
- Сравнивает типы объектов, проверяет ссылочную целостность и вызывает objA.Equals (objB)
- № 15 Выберите принципы ООП
- Инкапсуляция
- Полиморфизм
- Наследование
- Обфускация
- Рефлексия
- Структуризация
- № 16 Что сделает программа C#, выполнив следующий код: Console.WriteLine(«Hello, World!»)?
- Удалит все значения с Hello, World!
- Напишет Hello, World!
- Вырежет слово Hello, World! из всего текста
- Напишет на новой строчке Hello, World!
- № 17 Что такое CLR?
- Common Language Runtime – общезыковая среда исполнения
- C# Language Runtime – среда исполнения C#
- Cultural Language Reference – унифицированный языковой стандарт
- Common Learning Reference – единая справочная система по .NET
- № 18 Что такое BCL?
- Base Class Library – библиотека базовых классов .NET
- Base Common Language – язык общезыковой среды исполнения
- Basic C# Lingua – язык общезыковой среды исполнения
- Behavior Class Library – спецификация интерфейсов классов .NET

№ 19 Среди следующих вариантов неявных преобразований типов выберите все допустимые

Из float в double

Из decimal в float

Из double в float

Из float в decimal

№ 20 Какой тип переменной используется в коде в C#: `int value = 42`

Знаковое 32-бит целое

Знаковое

8-бит целое

1 байт*

Знаковое

64-бит целое

ОПК-8

Вопросы открытого типа:

№ 1 Как может быть создан объект некоторого класса в C++?

№ 2 Пусть имеется некоторый класс A. В программе на языке C++ есть строка:

`extern A a;`

Что происходит в данной строке?

№ 3 Дано объявление класса в C++:

`class Tovar`

`{`

`public:`

`char * nazva;`

`int price;`

`void Show ();`

`Tovar ();`

```
Tovar (char *, int);
```

```
};
```

Каково количество полей и методов этого класса?

- № 4 Что такое деструктор в C++?
- № 5 Что такое Класс в C++?
- № 6 Что такое перегрузка оператора в C++?
- № 7 Что такое переопределение операций в C++?
- № 8 STL расшифровывается как
- № 9 Базовый класс двунаправленного потока в C++
- № 10 Есть ли в представленной программе на языке C++ логические ошибки?

```
class Parent {
```

```
public:
```

```
    ~Parent() { }
```

```
    virtual void method() { }
```

```
};
```

```
class Child : public Parent {
```

```
public:
```

```
    Child() { /* захват ресурсов */ }
```

```
    ~Child() { /* освобождение ресурсов */ }
```

```
    void method() { /* программный код */ }
```

```
};
```



```

int main()

{

    Parent * obj = new Child;

    // программный код

    delete obj;

    return 0;

}

```

- № 11 Чем отличаются типы данных float, decimal и double в C#?
- № 12 Как определяется способ сортировки массива пользовательских типов с помощью метода Sort() в C#?
- № 13 Какой тип данных будет у переменной x в следующем случае: var x =6.3 в C#?
- № 14 Чем отличаются методы clone() и copy() при использовании с массивами в C#?
- № 15 В чем отличие методов Int64.Parse и Int64.TryParse в C#?
- № 16 В каком случае происходят коллизии при работе с объектом класса Dictionary?
- № 17 Какой алгоритм сортировки использует метод Sort класса List в C#?
- № 18 В чём принципиальное отличие ArrayList от List в C#?
- № 19 Какой уровень доступа по умолчанию у интерфейса в C#?
- № 20 Какое максимальное число блоков catch может быть использовано в комбинации с одним блоком try в C#?
- Вопросы закрытого типа:*
- № 1 Пусть класс Complex описывает комплексное число в C++. Оператор умножения перегружен как метод этого класса

```

class complex

```

```

{

```

```

    float re, im;

```

public:

complex operator*(const complex a);

};

Отметьте правильный способ вызова операторной функции для расчета значения $z3 = z1 * z2$ (где $z1, z2, z3$ - объекты класса `Complex`)

$z3 = z1 * z2;$

$z3 = \text{operator}*(z1, z2);$

$z3 = \text{operator}(z1 * z2);$

$z3.\text{operator}*(a, b);$

№ 2 По умолчанию все члены класса в C++ являются...

закрытыми

защищенными

открытыми

не определено

№ 3 Каким способом из предложенных корректно объявляется взаимная дружелюбность двух классов?

`class B; class F { friend class B; private: }; class B { friend class F; };`

`class B; class F { friend class B(F); }; class B { friend class F(B); };`

`class B; class F { friend class B friend class F; }; class B { friend class F friend class B; };`

`class B; class F { friend class B, F; }; class B { friend class F, B; };`

№ 4 Дан фрагмент программы на языке C++:

`class A { int Fx(); }`

`class B { friend class A; }`

```
class C {friend class B;}
```

Какие классы каким являются дружественными?

Все классы дружественны друг другу

Класс А является дружественным классу В, класс В является дружественным классу С, класс А не является дружественным классу С

Класс А является дружественным классу В, класс В является дружественным классу С, класс А является дружественным классу С

Класс С является дружественным классу В, класс В является дружественным классу А, класс А не является дружественным классу С

№ 5 Виртуальными в С++ называются функции (выберите все подходящие определения)

функции базового класса, которые могут быть переопределены в производном классе

функции базового класса, которые не используются в производном классе

функции базового класса, которые не могут быть переопределены в базовом классе

функции производного класса, переопределенные относительно базового класса

№ 6 Возможность и способ обращения производного класса к элементам базового в С++ определяется

ключами доступа: private, public, protected в заголовке объявления производного класса

ключами доступа: private, public, protected в теле производного класса

только ключом доступа protected в заголовке объявления производного класса

ключами доступа: private, public, protected в теле базового класса

№ 7 Какими могут быть функции по отношению к классу в С++?

дружественными

дружелюбными

дружескими

дружными

№ 8 Дружественная функция в C++ имеет доступ...

ко всем членам класса

только к открытым членам класса

только к закрытым членам класса

только к закрытым и защищенным членам класса

№ 9 Выберите верное утверждение

Функция может быть дружественной по отношению к нескольким классам

Все члены класса дружественны друг другу

Дружественные функции являются членами класса

Функция может быть дружественной только одному классу

№ 10 Дружественная функция – это

функция, объявленная в классе с атрибутом friend, но не являющаяся членом класса

функция другого класса, среди аргументов которой есть элементы данного класса

функция, являющаяся членом класса и объявленная с атрибутом friend

функция, которая в другом классе объявлена как дружественная данному

№ 11 В чём главное отличие объектов классов System.String и System.StringBuilder в C#?

System.String иммутабелен System.StringBuilder мутабелен

System.StringBuilder иммутабелен System.String мутабелен

System.String ковариантен System.StringBuilder контрвариантен

System.String потокобезопасен System.StringBuilder нет

№ 12 Какой из следующих способов организации строки в C# позволит использовать встроенные методы в случае необходимости поиска подстроки?

Используя System.String

Используя System.StringBuilder

Используя массив char

Используя указатель на char

- № 13 Объект какого из перечисленных классов используется для чтения информации из текстового потока в C#?
- StreamReader
- TextReader
- TextStream
- FileReader
- № 14 Вспомним предмет Структуры и организация данных. У коллекции Dictionary в C# следующие сложности операций:
- Добавление элемента [[1]]
- Удаление элемента [[2]]
- Поиск элемента [[3]]
- Укажите верные сложности из списка:
- O(1) или O(n) в случае коллизии
- O(1)
- O(n) или O(n²) в случае коллизии
- O(n)
- O(n²)
- № 15 Как устроена коллекция Dictionary в C#?
- Он содержит две коллекции int[] buckets и Entry[] entries.
- Он представляет собой чёрно/белое дерево
- Он представляет собой циклический двусвязный список
- Он содержит две коллекции Bucket[] buckets и T[] entries.
- № 16 Какой из указанных методов в C# позволяет (при корректных параметрах) добавить элемент в начало не пустого списка (List), а какой - в конец?
- Insert в начало, а Add в конец
- Add в начало, а Insert в конец
- AddToBegin в начало, AddToEnd в конец
- Begin в начало, End в конец
- Add в зависимости от переданного параметра

- № 17 Выберите верные ограничения к обобщениям (generics) в C#
- К обобщённому классу нельзя применять модификатор extern
 - К обобщённому типу не могут быть применены арифметические операции
 - типы указателей нельзя использовать в аргументах типа
 - Обобщённый класс не может содержать статические поля
 - Обобщённый тип не может быть value type
 - Обобщённый класс не может содержать виртуальные методы
- № 18 Применительно к языку C#:
- При неявной реализации в типе определяется метод с соответствующей сигнатурой [[1]] . Этот метод виден как обычный метод типа.
- При явной реализации в определении метода в типе [[2]].
- Тип [[3]] содержать одновременно явную реализацию интерфейса и метод с тем же именем и сигнатурой.
- Сопоставьте пропускам верные выражения из числа указанных ниже:
- без указаний на интерфейс
 - явно указывается имя интерфейса
 - может
 - не может
 - неявно указывается имя интерфейса
 - с указанием на интерфейс
- № 19 Возможно ли неявное преобразование класса к реализуемому им интерфейсу в C#?
- Да
 - Нет
- № 20 Для чего в .NET используется конструкция using(...){...}?
- Конструкция using позволяет вызывать метод Dispose интерфейса IDisposable автоматически, как только нужный объект выйдет за блок using.
 - Конструкция using просто сокращает запись вызова метода
 - Конструкция using позволяет вызывать метод Disposable интерфейса

IDisposable автоматически, как только нужный объект выйдет из области видимости программы

Конструкция using и IDisposable никак не связаны

ОПК-9

Вопросы открытого типа:

- № 1 Какие основные элементы STL решают вопросы обработки данных, размещённых в стандартных контейнерах STL?
- № 2 Как объявить дружественную функцию в C++?
- № 3 Какая конструкция C++ определена ниже?

```
template
```

```
T sum(T x, T y)
```

```
{
```

```
    T z = x + y;
```

```
    return z;
```

```
}
```

- № 4 Когда деструктор базового класса в C++ должен быть виртуальным?
- № 5 метод контейнера возвращающий итератор на первый элемент в C++
- № 6 Какой метод контейнера возвращает итератор на первый элемент в C++?

- № 7 метод контейнера возвращающий итератор на последний элемент в C++
- № 8 Наследование это
- № 9 Чем определяется, можно ли объекту производного класса в C++ обращаться к элементам базового класса и каким при этом должен быть способ обращения?
- № 10 Имеется следующее объявление классов в C++:

```
#include
```

```
using namespace std;
```

```

class Object

{

public:

    Object() { cout << "Object::ctor()" << endl; }

    ~Object() { cout << "Object::dtor()" << endl; }

};


class Base

{

public:

    Base() { cout << "Base::ctor()" << endl; }

    virtual ~Base() { cout << "Base::dtor()" << endl; }

    virtual void print() = 0;

};


class Derived: public Base

{

```


public:

```
Derived() { cout << "Derived::ctor()" << endl; }
```

```
~Derived() { cout << "Derived::dtor()" << endl; }
```

```
void print() {}
```

```
Object obj;
```

```
};
```

Объекты каких классов можно создать?

№ 11 Что представляют собой CLS-несовместимые исключения в C#?

№ 12 Что содержит поле TargetSite объекта исключения в C#?

№ 13 Сколько поколений объектов предусмотрено сборщиком мусора в C#?

№ 14 В каком режиме работает сборщик мусора по умолчанию в C#?

№ 15 Для каких типов поддерживается ковариантность и контравариантность у делегатов в C#?

№ 16 В чем различие делегатов Func и Action в C#?

№ 17 При компиляции класса, содержащего поле event, в C# в классе создаются:

№ 18 Что выведет на экран следующий фрагмент программы C#?

```
#define N 4
```

```
//.... Здесь подключение пространств имён и начало программы
```

```
#if N
```

```
Console.WriteLine(N);
```

```
#endif
```

```
Console.WriteLine(2);
```

№ 19 Что выведет на экран следующий фрагмент программы C#?

```
#define N
```

```
//... Здесь подключаются пространства имён и начинается программа
```

```
int N = 42;

Console.WriteLine(N);
```

№ 20 Что будет результатом выполнения программы C#?

```
using System;

class Program
{
    unsafe static void Main()
    {
        fixed (char* value = "sam")
        {
            char* ptr = value;
            while (*ptr != '\0')
            {
                Console.WriteLine(*ptr);
                ++ptr;
            }
        }
    }
}
```

Вопросы закрытого типа:
 № 1 к ассоциативным контейнерам в C++ относятся

set, map, multiset, multimap
 vector, list, deque, set, map
 vector, set, map
 vector, list, deque, set, map, string

№ 2 к последовательным контейнерам в C++ относятся

vector, list, deque
 vector, list, deque, set, map

vector, set, map

vector, list, deque, set, map, string

№ 3 Какие есть виды итераторов в C++?

Входные, выходные, однонаправленные, двунаправленные, произвольного доступа

Входные, выходные, многопоточные, двунаправленные

Однонаправленные, двумерные

Статические, константные, однонаправленные, двунаправленные

№ 4 Какие основные элементы STL абстрагируют перемещение по коллекциям объектов?

итераторы STL

обобщённые алгоритмы STL

контейнеры STL

функциональные объекты

№ 5 Какие основные элементы STL инкапсулируют хранение различных значений и объектов

контейнеры STL

обобщённые алгоритмы STL

итераторы STL

функциональные объекты

№ 6 Каким образом из перечисленных лучше хранить массив булевских значений

std::vector

std::vector

std::array

std::vector

№ 7 Какое утверждение про обобщённые алгоритмы в C++ верно?

обобщённые алгоритмы предоставляют обобщённое решение разных задач обработки данных в обобщённых контейнерах

для любого алгоритма существует копирующая версия алгоритма, которая переносит обработанные данные в другой контейнер

стандартом утверждена строгая классификация обобщённых алгоритмов

обобщённые алгоритмы упорядочивания не требуют определения операции сравнения элементов

№ 8 какой код позволяет переставить элементы в последовательности в C++

reverse()
replace()
move()
moveTo()

№ 9 какой код позволяет применить некоторую операцию ко всем элементам последовательности в STL

for_each()
forall()
while()
for_all()

№ 10 Выберите верное утверждение

В одной программе можно использовать шаблонные функции, созданные только по одному шаблону

По одному шаблону функции в одной программе может быть создана только одна шаблонная функция

По одному шаблону функции в одной программе может быть создано несколько разных шаблонных функций, отличающихся друг от друга только типами параметров

По одному шаблону функции может быть создано несколько разных шаблонных функций, отличающихся друг от друга типами и количеством параметров

- № 11 Верно ли следующее утверждение:
Методы интерфейса не имеют модификаторов доступа в C#
Верно
Не верно
- № 12 Некоторые исключения в C# являются прямыми потомками типа `[[1]]` . Некоторые CLR-исключения наследуются от типа `[[2]]`, а некоторые исключения приложений - от типа `[[3]]`

Сопоставьте пропускам верные выражения из числа указанных ниже:

`Exception`
`ApplicationException`
`SystemException`
`CLRException`
`SolutionException`
`System.Exceptions`
- № 13 Какие из следующих строчек содержат верный синтаксис условного блока `catch` в C#?
`catch(DivideByZeroException) when (b==0)`
`catch(Exception ex) when (ex is DivideByZeroException)`
`catch(DivideByZeroException) if (b==0)`
`catch(Exception ex == DivideByZeroException)`
- № 14 Какие из следующих конструкций допустимы в C#?
`try...catch`
`try..finally`
`try...`
`try..catch..finally`
`catch..finally`
- № 15 Для контроля или мониторинга времени жизни объекта в C# вызывается статический метод `[[1]]` объекта `[[2]]`, передающий ссылку на этот объект, и тип `[[3]]`, представляющий собой флаг, задающий способ контроля объекта.

Сопоставьте пропускам верные выражения из числа указанных ниже:

Alloc

GCHandle

GCHandleType

GC

TimeToLive

ControlType

№ 16 Выберите варианты, при которых наступает сборка мусора

Принудительный запуск GC.Collect()

ОС сообщает о нехватке памяти

Завершение работы CLR

Вызов у объекта метода Stop()

№ 17 Какие действия совершаются при обычной сборке мусора?

Маркирование, сжатие

Маркирование, удаление, перемещение, сжатие

Поиск и удаление

Просмотр таблицы дескрипторов, перемещение

№ 18 Верно ли следующее утверждение: Сборщик мусора при работе использует алгоритм отслеживания ссылок

Верно

Не верно

№ 19 Открытый статический метод [[1]] класса [[2]] в C# добавляет метод в цепочку делегатов.

Метод [[3]] сканирует массив делегатов (с конца и до члена с нулевым индексом), управляемый делегатом, на

который ссылается первый параметр. Он ищет делегат, поля [[4]] которого совпадают с соответствующими полями второго параметра.

Сопоставьте пропускам верные выражения из числа указанных ниже:

Combine

Delegate

Remove

Add

Delete

_target и _methodPtr

_chain и _pointer

_chain и _methodPtr

_pointer и _target

№ 20

[[1]] означает, что метод может вернуть тип, производный от типа, возвращаемого делегатом.

[[2]] означает, что метод может принимать параметр, который является базовым для типа параметра делегата.

Сопоставьте пропускам верные термины из числа указанных ниже:

Ковариантность

Контравариантность

Коваритивность

Конвариантивность

Иммутабельность

Мутабельность