Федеральное государственное бюджетное образовательное учреждение
высшего образования
Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

УТВЕРЖДАЮ
декан факультета вычислительной
математики и кибернетики

_____ /И.А. Соколов /
«27» сентября 2022г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

Языки программирования

Уровень высшего образования:

бакалавриат

Направление подготовки / специальность:

01.03.02 "Прикладная математика и информатика" (3++)

Направленность (профиль) ОПОП:

Искусственный интеллект и анализ данных

Форма обучения:

очная

Рассмотрен и утвержден
на заседании Ученого совета факультета ВМК
(протокол №7, от 27 сентября 2022 года)

Москва 2022

1. ФОРМЫ И ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

В процессе и по завершении изучения дисциплины оценивается формирование у студентов следующих компетенций:

Планируемые результаты обучения по дисциплине (модулю)		
Содержание и код компетенции.	**Индикатор (показатель) достижения компетенции**	**Планируемые результаты обучения по дисциплине, сопряженные с индикаторами достижения компетенций**
ОПК-5. Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-5.1. Разрабатывает программу для решения задачи с использованием языка высокого уровня. ОПК-5.2. Умение создавать, тестировать и отлаживать программы на языках программирования высокого уровня на компьютере. ОПК-5.3. Навыки написания качественного и хорошо документированного программного кода	Знать: 1. основные концепции и понятия современных языков программирования; 2. основные парадигмы программирования – процедурную, объектно-ориентированную и функциональную, иметь представление о логической парадигме программирования, понимать, как соотносятся конструкции и понятия конкретных языков программирования с парадигмами, поддерживаемыми этими языками, понимать отличия, достоинства и недостатки каждой парадигмы; 3. принципы проектирования, оценки и сравнения языков программирования; 4. основные технологические потребности современных технологий программирования, а также ключевые понятия и конструкции современных языков, реализующие эти технологические потребности; 5. основные точки зрения на индустриальные языки программирования и критерии оценки этих языков; 6. основные приемы и методы программирования в зависимости от используемых в языке программирования парадигм;

		Уметь: 1. применять на практике возможности современных языков программирования; 2. использовать механизмы абстракции, инкапсуляции и асинхронного программирования в современных языках программирования; 3. применять на практике концепции основных парадигм программирования – абстрактные классы и интерфейсы, замыкания, функции как объекты первого порядка, обработку исключительных ситуаций для повышения отказоустойчивости, асинхронные потоки и сопрограммы; 4. оценивать и сравнивать различные языки программирования с точки зрения эффективности использования вычислительных ресурсов, надежности, гибкости программирования, удобства сопровождения и модификации программ; 5. использовать на практике возможности и конструкции языков программирования, оптимально подходящие для решения задач, возникающих на разных этапах жизненного цикла программного продукта; 6. использовать на практике методы программирования из различных парадигм программирования, прежде всего объектно-ориентированной и функциональной; Владеть: 1. навыками работы с современными системами

		программирования на различных языках программирования; 2. навыками программирования в основных парадигмах программирования; 3. навыками оценки и сравнения языков программирования в зависимости от специфики решаемой задачи.

1.1. Текущий контроль успеваемости

Текущий контроль успеваемости осуществляется путем оценки результатов выполнения заданий практических (семинарских) занятий, самостоятельной работы, предусмотренных учебным планом и посещения занятий/активность на занятиях.

В качестве оценочных средств текущего контроля успеваемости предусмотрены:

контрольная работа

Пример контрольной работы

Вариант 1

1. Замените знаки вопроса (?????) в приведенной ниже программе на языке C# так, чтобы в стандартный вывод выдавалась сумма квадратов эдементов списка s. Никакой код куда-либо еще добавлять нельзя. Примечание: метод ForEach списка имеет прототип `List<int>.ForEach(Action<int> action)` и выполняет действие action для каждого элемента списка.

```
List<int> l = new List<int>() { 1, 2, 3, 5, 7, 9 };
int s = 0;
l.ForEach(?????);
System.Console.WriteLine(s);
```

2. Пусть s описана как переменная строкового типа (string), и в программе есть присваивание s= ”кафе” (все буквы - кириллические). Какова будет длина этой строки в языках C++(s.size()), Java (s.length), Go (len(s)), Swift (s.count)? Для каждого языка дайте ответ и объясните различия (если они есть).

3. Объясните, допустимо ли присваивание aobj = afoo в следующем фрагменте программы на Java. Если допустимо, то какие проблемы с aobj и afoo могут возникнуть в дальнейшем при выполнении программы?

```
class Foo { public void foo () {} }
class Bar { public void bar () {} }
Object [] aobj; Foo [] afoo = new Foo[10];
aobj = afoo;
```

4. Написать на языке C++ шаблонную функцию int MakeWord(), которая возвращает целое число, являющееся побитовой суммой (операция суммирования — побитовое «или» |) аргументов шаблона. Аргументы шаблона должны быть целочисленными.

Например std::cout << MakeWord<1,2>(); должно выдать 3, а

std::cout << MakeWord<2,4,9>(); должно выдать 15.

5. Являются ли реализации обобщенных классов на языке Java более эффективными, чем их необобщенные аналоги. Например, будет ли реализация обобщенного класса Stack<int> более эффективной, чем реализация класса Stack из Object?

6. Напишите какой-нибудь пример генератора (но не сопрограммы) на языке Python. Чем сопрограммы языка Python отличаются от простых генераторов в этом языке?

7. Дайте определение вырезки из массива. Приведите примеры вырезок в языке Python и Go. Чем они отличаются?

8. В приведенной ниже программе на языке Go есть ошибка, которая проявляется во время выполнения. В чем она заключается, и как ее исправить (указание — надо переставить одну строку в другое место)? Что будет выдано после исправления?

```
package main; import "fmt"
func fibonacci(c, quit chan int) {
x, y := 0, 1
for {
     select {
     case c <- x:
          x, y = y, x+y
     case <-quit:
          fmt.Println("quit")
          return
     }
}
}

 func main() {
     c := make(chan int)
     quit := make(chan int)
     go func() {
       for i := 0; i < 10; i++ {
          fmt.Print(←c);fmt.Print(" ")
       }
     }()
     quit <- 0
     fibonacci(c, quit)
}
```

Вариант 2

1. Замените знаки вопроса (?????) в приведенной ниже программе на языке C# так, чтобы в стандартный вывод выдавалась сумма квадратов эдементов списка s. Никакой код куда-либо еще добавлять нельзя. Примечание: метод ForEach списка имеет прототип List<int>.ForEach(Action<int> action) и выполняет действие action для каждого элемента списка.

```
List<int> l = new List<int>() { 1, 2, 3, 5, 7, 9 };
int s = 0;
l.ForEach(?????);
System.Console.WriteLine(s);
```

2. Пусть s описана как переменная строкового типа (string), и в программе есть присваивание s= ”кафе” (все буквы - кириллические). Какова будет длина этой строки в языках C++(s.size()), Java (s.length), Go (len(s)), Swift (s.count)? Для каждого языка дайте ответ и объясните различия (если они есть).

3. Объясните, допустимо ли присваивание aobj = afoo в следующем фрагменте программы на Java. Если допустимо, то какие проблемы с aobj и afoo могут возникнуть в дальнейшем при выполнении программы?

```
class Foo { public void foo () {} }
class Bar { public void bar () {} }
Object [] aobj; Foo [] afoo = new Foo[10];
aobj = afoo;
```

4. Написать на языке C++ шаблонную функцию int MakeWord(), которая возвращает целое число, являющееся побитовой суммой (операция суммирования — побитовое «или» |) аргументов шаблона. Аргументы шаблона должны быть целочисленными.

Например std::cout << MakeWord<1,2>(); должно выдать 3, а
std::cout << MakeWord<2,4,9>(); должно выдать 15.

5. Являются ли реализации обобщенных классов на языке Java более эффективными, чем их необобщенные аналоги. Например, будет ли реализация обобщенного класса Stack<int> более эффективной, чем реализация класса Stack из Object?

6. Напишите какой-нибудь пример генератора (но не сопрограммы) на языке Python. Чем сопрограммы языка Python отличаются от простых генераторов в этом языке?

7. Дайте определение вырезки из массива. Приведите примеры вырезок в языке Python и Go. Чем они отличаются?

8. В приведенной ниже программе на языке Go есть ошибка, которая проявляется во время выполнения. В чем она заключается, и как ее исправить (указание — надо переставить одну строку в другое место)? Что будет выдано после исправления?

```
package main; import "fmt"
func fibonacci(c, quit chan int) {
    x, y := 0, 1
    for {
        select {
        case c <- x:
            x, y = y, x+y
        case <-quit:
            fmt.Println("quit")
            return
        }
    }
}

func main() {
    c := make(chan int)
    quit := make(chan int)
    go func() {
      for i := 0; i < 10; i++ {
        fmt.Print(←c);fmt.Print(" ")
      }
    }()
    quit <- 0
    fibonacci(c, quit)
}
```

1.2. Промежуточная аттестация

Промежуточная аттестация осуществляется в форме экзамена
В качестве средств, используемых на промежуточной аттестации предусматривается: Билеты

1.3. Типовые задания для проведения промежуточной аттестации

Вопросы к экзамену.

1. Основные позиции при рассмотрении ЯП. Схема рассмотрения ЯП: базис, средства развития и средства защиты.
2. Основные понятия языков программирования: данные, операции и связывание.
3. Понятие о виртуальной машине языка
4. Классификация базисных скалярных типов данных
5. Арифметические типы данных: целые, плавающие, фиксированные. Проблемы представления чисел и способы их решения в ЯП.
6. Символьные и логические типы данных.
7. Порядковые типы: диапазоны и перечисления. Особенности реализации перечислений в современных ЯП.
8. Ссылки и указатели. Управление памятью. Автоматическая сборка мусора. Объектно-референциальная модель в современных ЯП.
9. Составные типы данных. Массивы и их особенности в современных ЯП. Динамические массивы и списки. Вырезки из массивов.
10. Записи. Недостатки системы типов в традиционных ЯП. Объединения как средство преодоления этих недостатков. Проблемы, связанные с объединениями.
11. Ассоциативные массивы и записи.
12. Понятие о структурном программировании. Разновидности управляющих конструкций в современных языках программирования. Оператор перехода, связанные с им проблемы и способы их решения в современных ЯП.
13. Условные операторы и многовариантные развилки. Циклы. Особенности реализации циклов-итераторов в современных ЯП
14. Статический полиморфизм и перегрузка имен подпрограмм.
15. Функции и функциональные типы. Функции как объекты первого порядка. Понятие замыкания и анонимные функции. Лямбда-функции и замыкания в современных ЯП (C#, Python, C++, Java, JavaScript).
16. Сопрограммы и подпрограммы. Сопрограммы в современных ЯП (на примере итераторов в C# и генераторов Python). Го-программы в языке Go.
17. Понятие логического модуля. Использование модулей для определения новых типов данных. Особенности понятия модуля в современных ЯП.
18. Импорт и экспорт имен. Видимость имен: непосредственная и потенциальная. Управление видимостью. Области видимости и пространства имен. Модульность и технология программирования: проектирование «сверху-вниз» и «снизу-вверх».
19. Понятие класса. Класс как тип данных. Члены класса: функции, данные. Статические и нестатические члены. Члены - вложенные классы. Статические и нестатические классы. Классы и области видимости.
20. Понятие специальных функций-членов. Проблема инициализации объектов и способы ее решения. Конструкторы, деструкторы, операторы using и try-finally.
21. Преобразование типов и классы. Явные и неявные преобразования. Управление преобразованиями в современных ЯП: проблемы и способы их решения.
22. Классы и перегрузка имен. Перегрузка встроенных знаков операций. Итераторы и индексаторы. Классы и стандартные библиотеки. Встроенные классы стандартной библиотеки.

23. Понятие инкапсуляции. Понятие абстрактного типа данных (АТД) и его достоинства. Инкапсуляция и логические модули.
24. Реализация АТД в модульных языках программирования (Ада, Оберон, Модула). Инкапсуляция и классы. Управление видимостью и управление доступом.
25. Пространства имен и инкапсуляция. Реализация АТД с помощью понятия класса. Принцип разделения определения, реализации и использования (РОРИ). Эволюция принципа РОРИ в современных ЯП.
26. Модульность и раздельная трансляция.
27. Исключительные ситуации и обработка ошибок.
28. Иерархии типов, статические и динамические типы в объектно-ориентированных ЯП.
29. Наследование и области видимости имен. Замещение, перегрузка и скрытие имен при наследовании. Наследование и инкапсуляция. Управление видимостью и доступом при наследовании. Запрещение наследования для классов и методов.
30. Наследование и специальные функции. Понятие о множественном наследовании.
31. Статическое и динамическое связывание методов. Динамический тип данных и динамическое связывание. Замещение функций и динамическое связывание. Особенности динамического связывания в современных ЯП. Достоинства и недостатки динамического связывания. Снятие динамического связывания.
32. Понятие о мультиметодах.
33. Понятие абстрактного класса (АК). Необходимость понятия АК при проектировании иерархий классов. Воплощение концепции АК в современных ЯП. Абстрактные классы и интерфейсы. Интерфейс как языковая конструкция.
34. Связь интерфейсов и других языковых конструкций (итераторов, сохраняемых объектов и т.д.). Интерфейсы и иерархии классов. Множественное наследование интерфейсов. Реализация интерфейсов и ее особенности современных ЯП. Явная и неявная реализация интерфейсов.
35. Функциональные интерфейсы и функциональные типы данных.
36. Понятие о динамической идентификации типа
37. Понятие о статической параметризации и родовых объектах. Достоинства статической параметризации. Статическая параметризация и ООП. Родовые модули и подпрограммы в языке Ада.
38. Механизм шаблонов в языке Си++. Шаблоны-классы и шаблоны-функции. Параметры шаблонов. Вывод параметров шаблонов. Генерация кода по шаблонам. Полная и частичная специализация шаблонов. Обобщенное программирование на языке Си++. Шаблоны и функциональное программирование.
39. Особенности обобщенного программирования в языках C# и Java.
40. Ковариация и обобщенные конструкции. Реализация ковариации в языках C# и Java.

Типовые задания для экзамена.

1. Написать на конкретном языке программирования объявления, функционально эквивалентные приведенным объявлениям на другом языке.
2. Найти возможные ошибки в приведенном фрагменте программы, если есть, то исправить их и объяснить, что будет выдано в стандартный канал вывода.
3. Дать определение и примеры какой-либо конструкции (исключительной ситуации, абстрактного класса, виртуальной функции, динамического связывания и т.д.).
4. Перечислить языки (из заданного списка), в которых есть некоторое понятие. Для каждого языка дать пример объявления и использования данного понятия.
5. Сравнить реализации некоторого понятия в двух конкретных языках программирования с некоторой точки зрения (эффективности реализации, удобства программирования, надежности, сложности использования, простоты понимания и т.п.

6. В заданном фрагменте программы внести (небольшие) изменения так, чтобы фрагмент выполнил определенную задачу, например, вывод заданной строки в стандартный канал вывода.
7. Переписать на некотором языке программирования заданный фрагмент программы на другом языке программирования.
8. Дать определение и объяснить назначение какой-либо конструкции в некотором языке программирования. Есть аналоги этой конструкции в языках программирования из приведенного списка. Если есть, то дать короткий пример объявления и использования для каждого языка.

Пример экзаменационного билета

1. Понятие инкапсуляции. Понятие абстрактного типа данных (АТД) и его достоинства. Инкапсуляция и логические модули.
2. Функциональные интерфейсы и функциональные типы данных.
3. Написать на конкретном языке программирования объявления, функционально эквивалентные приведенным объявлениям на другом языке.

2. КРИТЕРИИ ОЦЕНКИ ПО ДИСЦИПЛИНЕ

ШКАЛА И КРИТЕРИИ ОЦЕНИВАНИЯ результатов обучения (РО) по дисциплине				
Оценка виды оценочных средств	**2 (не зачтено)**	**3 (зачтено)**	**4 (зачтено)**	**5 (зачтено)**
Знания **(виды оценочных средств: приведены в п. 1.2.)**	Отсутствие знаний	Фрагментарные знания	Общие, но не структурированные знания	Сформированные систематические знания
Умения **(виды оценочных средств: приведены в п. 1.2.)**	Отсутствие умений	В целом успешное, но не систематическое умение	В целом успешное, но содержащее отдельные пробелы умение (допускает неточности непринципиального характера)	Успешное и систематическое умение
Навыки **(владения, опыт деятельности)** **(виды оценочных средств: приведены в п. 1.2..)**	Отсутствие навыков (владений, опыта)	Наличие отдельных навыков (наличие фрагментарного опыта)	В целом, сформированные навыки (владения), но используемые не в активной форме	Сформированные навыки (владения), применяемые при решении задач