МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Факультет информационных технологий

**УТВЕРЖДАЮ**
Декан факультета
«Информационные технологии»
_____/Д.Г.Демидов/
«28» июня 2021

# РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

## «Методологии программирования»

Направление подготовки/специальность
**09.03.03 Прикладная информатика**

Профиль/специализация
**«Корпоративные информационные системы»**

Квалификация
**бакалавр**

Формы обучения
**очная**

Москва, 2021 г.

# 1 Цели, задачи и планируемые результаты обучения по дисциплине

Цель преподавания дисциплины

Формирование у студентов совокупности систематизированных знаний о моделях объектов профессиональной деятельности, реализуемых на основе прикладных информационных средств и технологий.

Задачи изучения дисциплины

Овладение информацией о моделях и методах, используемых при проектных и исследовательских работах в области профессиональной деятельности;

получение навыков применения современных моделей и методов при

решении задач профессиональной деятельности;

формирование умения использовать программные средства автоматизации проектных и исследовательских работ.

Обучение по дисциплине направлено на формирование у обучающихся следующих компетенций:

| **Код и наименование компетенций** | **Индикаторы достижения компетенции** |
|---|---|
| ПК-3. Способен разрабатывать требования и проектировать программное обеспечение | ИПК-3.1. Знает возможности существующей программно-технической архитектуры; возможности современных и перспективных средств разработки программных продуктов, технических средств; методологии разработки программного обеспечения и технологии программирования; методологии и технологии проектирования и использования баз данных; языки формализации функциональных спецификаций; методы и приемы формализации задач; методы и средства проектирования программного обеспечения; методы и средства проектирования программных интерфейсов; методы и средства проектирования баз данных; принципы построения архитектуры программного обеспечения и виды архитектуры программного обеспечения; типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке программного обеспечения; методы и средства проектирования программного обеспечения и баз данных; методы и средства проектирования программных интерфейсов. |

| | |
|---|---|
| | ИПК-3.2. Умеет проводить анализ исполнения требований; вырабатывать варианты реализации требований; проводить оценку и обоснование рекомендуемых решений; осуществлять коммуникации с заинтересованными сторонами; выбирать средства реализации требований к программному обеспечению; вырабатывать варианты реализации программного обеспечения; проводить оценку и обоснование рекомендуемых решений; осуществлять коммуникации с заинтересованными сторонами; использовать существующие типовые решения и шаблоны проектирования программного обеспечения; применять методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов; осуществлять коммуникации с заинтересованными сторонами.<br>ИПК-3.3. Владеет современным инструментарием и средами разработки программного кода; современным инструментарием и средами проектирования программного кода, методами тестирования ПО. |

## 2 Место дисциплины в структуре образовательной программы

Дисциплина относится к части, формируемой участниками образовательных отношений блока Б1.2 и междисциплинарно связана с поддерживающими дисциплинами: Основы разработки корпоративных информационных систем, Прикладное программирование и последующими дисциплинами: Архитектура автоматизированных систем.

## 3 Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зачетных единицы (144 часа).

Дисциплина ведется на втором курсе в третьем семестре.

Из них 72 часа аудиторные (практические занятия) и 72 часа самостоятельной работы студентов. Форма итоговой аттестации зачет.

### 3.1 Тематический план изучения дисциплины для очной формы обучения

3.2.1 Очная форма обучения

| № п/п | Разделы/темы дисциплины | Трудоемкость, час | | |
|---|---|---|---|---|
| | | Всего | Аудиторная работа | |

| | | | Лекции | Семинарские/ практические занятия | Лабораторные занятия | Практическая подготовка | **Самостоятельная работа** |
|---|---|---|---|---|---|---|---|
| 1 | Понятие программирования; Программная инженерия | 18 | | | 9 | | 9 |
| 2 | Интегрированные среды программирования | 18 | | | 9 | | 9 |
| 3 | Методологии программирования, настройка устройств; Машинные и машинно-ориентированные языки | 18 | | | 9 | | 9 |
| 4 | Языки программирования высокого уровня | 18 | | | 9 | | 9 |
| 5 | Структурное программирование; Процедурноориентированное программирование | 18 | | | 9 | | 9 |
| 6 | Объектно-ориентированное программирование | 18 | | | 9 | | 9 |
| 7 | Объекты задач и объекты программ | 18 | | | 9 | | 9 |
| 8 | Проектирование архитектуры программной системы | 18 | | | 9 | | 9 |
| **Итого** | | **144** | | | **72** | | **72** |

### 3.2 Содержание дисциплины

Понятие программирования; Программная инженерия

Интегрированные среды программирования;

Специализированный текстовый редактор

Методологии программирования, настройка устройств; Машинные и машинно-ориентированные языки

Основные схемы преобразования исходного модуля в исполняемый модуль; Статически и динамически компонуемые библиотеки; Составные части системы программирования

История программирования, первые автоматические вычислители; Возникновение языков программирования

Языки программирования высокого уровня;

Достоинства и недостатки языков высокого уровня

Структурное программирование; Процедурноориентированное программирование

Объекты, атрибуты, методы; Разработка объектов

Инкапсуляция, полиморфизм, наследование

Объекты задач и объекты программ; Постановка задачи при разработке программ

Проектирование архитектуры программной системы; Методы проектирования программ, ориентированные на обработку

Методы проектирования программ, ориентированные на использование структур данных

Моделирование при решении задач программирования; Разработка алгоритма; Процесс программирования

Каскадная схема жизненного цикла программы;

Итерационная схема жизненного цикла программы

### 3.3 Тематика лабораторных занятий

Лабораторная работа 1. Знакомство с окружением для разработки

Лабораторная работа 2. Дифференцирование функций

Лабораторная работа 3. Вычисление определённых интегралов

Лабораторная работа 4. Решение нелинейного уравнения

Лабораторная работа 5. Нахождение экстремума функции

Лабораторная работа 6. Векторы, матрицы и системы линейных алгебраических уравнений

Лабораторная работа 7. Решение дифференциальных уравнений и систем дифференциальных уравнений

Лабораторная работа 8. Интерполяция и экстраполяция функций

Лабораторная работа 9. Математическая обработка экспериментальных данных

## 4 Учебно-методическое и информационное обеспечение

### 4.1 Основная литература

1. Технологии программирования : учебное пособие / А. В. Гайдель, А. В. Благов, В. И. Проценко, А. С. Широканев. — Самара : Самарский университет, 2020. — 108 с. — ISBN 978-5-7883-1554-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/189025 (дата обращения: 28.09.2021). — Режим доступа: для авториз. пользователей.

### 4.2 Дополнительная литература

Семкин, А. О. Информационные технологии. Общие вопросы информатики, алгоритмизации и программирования : учебное пособие / А. О. Семкин, А. С. Перин. — Москва : ТУСУР, 2020. — 163 с. — ISBN 978-5-86889-898-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/313442 (дата обращения: 28.09.2021). — Режим доступа: для авториз. пользователей.

### 4.3 Программное обеспечение

1. Visual Studio Code
2. Браузеры Chrome, Edge, Firefox
3. OpenVPN с правами для запуска у студентов
4. FileZilla
5. PuTTy
6. Git
7. Node.js 18
8. Python 3.10
9. Wireshark
10. Программа MathCAD

## 5 Материально-техническое обеспечение

Для проведения лабораторных работ и самостоятельной работы студентов подходят аудитории, оснащенные компьютерами с программным обеспечением в соответствии со списком в пункте 4.5 и подключенные к интернету.

Число рабочих мест в аудитории должно быть достаточным для обеспечения индивидуальной работы студентов.

Рабочее место преподавателя должно быть оснащено компьютером с подключенным к нему проектором или иным аналогичным по функциональному назначению оборудованием.

# 6 Методические рекомендации

## 6.1 Методические рекомендации для преподавателя по организации обучения

1. При подготовке к занятиям следует предварительно проработать материал занятия, предусмотрев его подачу точно в отведенное для этого время занятия. Следует подготовить необходимые материалы – теоретические сведения, задачи и др. При проведении занятия следует контролировать подачу материала и решение заданий с учетом учебного времени, отведенного для занятия.

2. При проверке работ и отчетов следует учитывать не только правильность выполнения заданий, но и оптимальность выбранных методов решения, правильность выполнения всех его шагов.

## 6.2 Методические указания для обучающихся по освоению дисциплины

Изучение дисциплины осуществляется в строгом соответствии с целевой установкой в тесной взаимосвязи учебным планом. Основой теоретической подготовки студентов являются лекции и самостоятельная работа.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, готовятся к промежуточной аттестации, а также самостоятельно изучают отдельные темы учебной программы.

На занятиях студентов, в том числе предполагающих практическую деятельность, осуществляется закрепление полученных, в том числе и в процессе самостоятельной работы, знаний. Особое внимание обращается на развитие умений и навыков установления связи положений теории с профессиональной деятельностью будущего специалиста.

Самостоятельная работа осуществляется индивидуально. Контроль самостоятельной работы организуется в двух формах:

- самоконтроль и самооценка студента;
- контроль со стороны преподавателей (текущий и промежуточный).

Текущий контроль осуществляется на аудиторных занятиях.

Критериями оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умения студента использовать теоретические знания при выполнении практических задач;
- сформированность компетенций;
- оформление материала в соответствии с требованиями.

Приветствуется обсуждение самих заданий с другими студентами: можно как давать, так и получать советы по общей стратегии выполнения и изучения материала, давать и получать помощь в отладке. Однако писать код студент должен самостоятельно. Делиться кодом или писать его совместно запрещено.

# 7 Фонд оценочных средств

## 7.1 Методы контроля и оценивания результатов обучения

Работа на лекции

Лекционные занятия включают изложение, обсуждение и разъяснение основных направлений и вопросов изучаемой дисциплины, знание которых необходимо в ходе реализации всех остальных видов занятий и в самостоятельной работе студентов. На лекциях студенты получают самые необходимые знания по изучаемой проблеме. Непременным

условием для глубокого и прочного усвоения учебного материала является умение студентов сосредоточенно слушать лекции, активно, творчески воспринимать излагаемые сведения. Внимательное слушание лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, конспектирование их помогает усвоить материал. Конспект является полезным тогда, когда записано самое существенное, основное. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях. Конспект лучше подразделять на пункты, параграфы, соблюдая красную строку. Принципиальные места, определения, формулы следует сопровождать замечаниями. Работая над конспектом лекций, всегда следует использовать не только основную литературу, но и ту литературу, которую дополнительно рекомендовал лектор.

Практические занятия

Подготовку к практическому занятию следует начинать с ознакомления с лекционным материалом, с изучения плана практических занятий. Определившись с проблемой, следует обратиться к рекомендуемой литературе. Владение понятийным аппаратом изучаемого курса является необходимым, поэтому готовясь к практическим занятиям, студенту следует активно пользоваться справочной литературой: энциклопедиями, словарями и др. В ходе проведения практических занятий, материал, излагаемый на лекциях, закрепляется, расширяется и дополняется при подготовке сообщений, рефератов, выполнении тестовых работ. Степень освоения каждой темы определяется преподавателем в ходе обсуждения ответов студентов.

Самостоятельная работа

Студент в процессе обучения должен не только освоить учебную программу, но и приобрести навыки самостоятельной работы. Самостоятельная работа студентов играет важную роль в воспитании сознательного отношения самих студентов к овладению теоретическими и практическими знаниями, привитии им привычки к направленному интеллектуальному труду. Самостоятельная работа проводится с целью углубления знаний по дисциплине. Материал, законспектированный на лекциях, необходимо регулярно дополнять сведениями из литературных источников, представленных в рабочей программе. Изучение литературы следует начинать с освоения соответствующих разделов дисциплины в учебниках, затем ознакомиться с монографиями или статьями по той тематике, которую изучает студент, и после этого – с брошюрами и статьями, содержащими материал, дающий углубленное представление о тех или иных аспектах рассматриваемой проблемы. Для расширения знаний по дисциплине студенту необходимо использовать Интернет-ресурсы и специализированные базы данных: проводить поиск в различных системах и использовать материалы сайтов, рекомендованных преподавателем на лекционных занятиях.

Подготовка к сессии

Основными ориентирами при подготовке к промежуточной аттестации по дисциплине являются конспект лекций и перечень рекомендуемой литературы. При подготовке к сессии студенту следует так организовать учебную работу, чтобы перед первым днем начала сессии были сданы и защищены все практические работы. Основное в подготовке к сессии – это повторение всего материала курса, по которому необходимо пройти аттестацию. При подготовке к сессии следует весь объем работы распределять равномерно по дням, отведенным для подготовки, контролировать каждый день выполнения работы.

## 7.2 Шкала и критерии оценивания результатов обучения

Критерии оценки выполнения тестового задания

Оценка Критерии оценивания

Неудовлетворительно от 0% до 30% правильных ответов из общего числа тестовых заданий

Удовлетворительно от 31% до 50% правильных ответов из общего числа тестовых заданий

Хорошо от 51% до 80% правильных ответов из общего числа тестовых заданий

Отлично от 81% до 100% правильных ответов из общего числа тестовых заданий

Критерии выполнения лабораторной работы

Неудовлетворительно Работа выполнена не полностью и объем выполненной части работы не позволяет сделать правильных выводов

Удовлетворительно Работа выполнена не полностью, но не менее 50% объема, что позволяет получить правильные результаты и выводы; в ходе проведения работы были допущены ошибки

Хорошо Работа выполнена в полном объеме с соблюдением необходимой последовательности действий, но допущена одна ошибка или не более двух недочетов и обучающийся может их исправить самостоятельно или с небольшой помощью преподавателя

Отлично Работа выполнена в полном объеме без ошибок с соблюдением необходимой последовательности действий

Критерии оценивания на зачете

Уровень1.

Недостаточный Незнание значительной части программного материала, неумение даже с помощью преподавателя сформулировать правильные ответы на задаваемые вопросы, невыполнение практических заданий Неудовлетворительно/Незачтено

Уровень 2.

Базовый Знание только основного материала, допустимы неточности в ответе на вопросы, нарушение логической последовательности в изложении программного материала, затруднения при решении практических задач Удовлетворительно/зачтено

Уровень 3.

Повышенный Твердые знания программного материала, допустимые несущественные неточности при ответе на вопросы, нарушение логической последовательности в изложении программного материала, затруднения при решении практических задач Хорошо/зачтено

Уровень 4.

Продвинутый Глубокое освоение программного материала, логически стройное его изложение, умение связать теорию с возможностью ее применения на практике, свободное решение задач и обоснование принятого решения Отлично/зачтено

## 7.3 Оценочные средства

7.3.1 Текущий контроль

Тест

Вопрос №1 .

Ключевое слово template предназначено в языке C++ для:

Варианты ответов:

1. работы с управляемой кучей
2. передачи функции адреса объекта
3. обеспечение возврата из функции
4. создания родовой функции

Вопрос №2 .

Приведено объявление класса. Укажите случай нарушения доступа при обращении к члену класса

class A

{

int x; public: void set(int); int get(); void show();

}; A a;

Варианты ответов:

1. a.set(7);
2. a.x;
3. a.get();
4. a.show();

Вопрос №3 .

Специальный указатель, который автоматически передается любой функции-члену при ее вызове и указывает на объект, генерирующий вызов

Варианты ответов:

1. extern
2. this
3. throw
4. explicit

Вопрос №4 .

Какая из операций для объектов (экземпляров классов) поддерживается компилятором по умолчанию Варианты ответов:

1. +
2. 3. =

4. ==

Вопрос №5 .

Если в классе A перегружается операция = =, какое должно быть возвращаемое значение функции, осуществляющей перегрузку

Варианты ответов:

1. A
2. bool 3. int

4. void

Примерный список вопросов

РАЗРАБОТКА КЛАССА СТРОКА

1. Цель работы

Разработать класс String, предоставляющий более удобные и безопасные средства для операций со строками нежели встроенный тип char*.

2. Пример конструирования класса

Основным недостатком встроенного типа является отсутствие контроля за выходом за границы массива символов, что может приводить к катастрофическим последствиям во время выполнения программы. Например: const SIZE = 80; char* str = new char[SIZE];

//. . .

str[SIZE] = '\0'; приводит к попытке записи в область памяти занятую совершенно другим объектом.

Определение класса может выглядеть следующим образом: class String{ public:

String(int);

String(char*); ~String(); private: int len; char* str;

};

Первый конструктор может быть применен для использования строки как буфера заданного размера:

String buf(1024); String::String(int ln) { len = ln;

str = new char[len+1]; str[0] = 0x00;

}

Второй конструктор создает объект типа строка используя в качестве аргумента встроенный тип char* адрес стандартной строки: String::String(char* s) { len = strlen(s); str = new char[len+1]; strcpy(str,s);

}

Здесь использованы функции стандартной библиотеки strlen(), strcpy() для определения длины строки и копирования строк, поэтому в файле содержащем тело конструктора перед вызовом этих функций должен быть подключен заголовочный файл string.h с помощью директивы:

#include <string.h>

Теперь в программе могут быть созданы и использоваться наравне со встроенными типами объекты String: char s[] = "Строка";

String s1(512);

String s2(s);

Компилятор C++ встретив вторую и третью строки приведенного фрагмента вызовет соответственно первый и второй конструктор String в соответствии с типами аргументов. Объекты s1, s2 перед выходом из своей области существования должны быть удалены из памяти деструктором класса String. Вызов деструктора осуществляется автоматически компилятором, задача же программиста предоставить соответствующий деструктор: String::~String(){ delete [] str;

}

Поскольку при создании объектов типа строка нами выделялась память под массив символов с помощью операции new деструктор должен содержать оператор освобождения памяти delete . Указание квадратных скобок необходимо для того чтобы информировать компилятор о том, что указатель str настроен на массив и необходимо освободить память выделенную под него.

Для того чтобы сделать безопасной работу с объектами типа String необходимо встроить в него проверку правильности обращения по индексу к элементам массива. Это можно сделать определив оператор индексации []. Дадим более полное определение класса:

class String{ public:

String(int);

String(char*);

~String();

char& operator [] (int); private: int len; char* str; };

Тело оператора индексации может выглядеть следующим образом: const int ERR_RANGE = -1;

#include <iostream.h> #include <stdlib.h>

char& String::operator[](int index) { if(index < 0 || index >= len) {

cerr << "Index out of bounds for String\n"; exit(ERR_RANGE);

}

return str[index];

}

В случае обращения к элементу строки по неправильному индексу, с помощью стандартного потока вывода ошибок, на экран выводится сообщение о выходе индекса за границы строки и с помощью стандартной функции exit(), описанной в stdlib.h передается управление операционной системе с извещением ее о коде ошибки.

3. Задание

Разработать класс String определив для него методы:

копирования строк, реализовав оператор = ; поиска подстроки;

слияния строк, реализовав операторы += и + ; эквивалентности строк, набор операторов == и !=; определения длины строки; вывода в поток, << ; ввода из потока >> ;

вставки подстроки с нужной позиции; конструктор копирования вида X::X(const X&);

Напишите программу иллюстрирующую основные методы класса String.

7.3.2 Промежуточная аттестация

1. Программа. Характеристика программ.
2. Понятие программного обеспечения. Виды программного обеспечения.
3. Программный продукт. Характерные особенности программного продукта.
4. Программный комплекс.
5. Программное средство. Определение требований к программному средству.
6. Программная система. Сложность и сложные системы. Источники сложности.
7. Признаки работоспособной сложной системы. Классификация программных систем по сложности.
8. Проблемы проектирования сложных программных средств.
9. Жизненный цикл программного обеспечения.
10. Управление проектом, планирование и распределение ресурсов, контроль исполнения сроков.
11. Тестирование и оценка качества.
12. Управление программными конфигурациями.
13. Сопровождение.
14. Модернизация и масштабирование программного обеспечения.

Тема 3. Стандарты и методики, используемые при разработке программных средств

15. Виды стандартов.
16. Методики проектирования.
17. Стандартизация жизненного цикла программного средства.
18. Понятие методологии.
19. Атрибуты методологий.
20. Основные методологии программирования.
21. Общая система понятий технологии программирования.
22. Технология. Процесс. Стадия. Технологический подход.
23. Критерии оценки технологий.

Тема 6. Проектирование библиотек классов

24. Виды классов: конкретный тип, абстрактный тип, узловой класс, интерфейсный класс.
25. Динамическая идентификация типа.

26. Управление видимостью и областью действия имен.
27. Управление памятью.
28. Библиотеки контейнерных классов.
29. Номенклатура контейнеров и примеры их использования.
30. Иерархия классов исключений.

Тема 7. Проектирование интерфейса с пользователем

31. Библиотеки интерфейсных элементов.
32. Понятие приложения.
33. Диалоговые окна и дочерние элементы управления.
34. Языки программирования четвертого поколения, CASE-системы, системы ускореннойразработки приложений.
35. Системный анализ.
36. Принципы объектно-ориентированного анализа и их обсуждение.
37. Язык объектного моделирования UML.
38. Основные определения: система, домен, подсистема, элемент, связи, среда.
39. Структура системы, декомпозиция, иерархия элементов.
40. Процессы в системе и потоки информации.
41. Исследование действий.
42. Построение моделей доменов и подсистем, связей и взаимодействия подсистем, взаимодействияобъектов, событий, процессов, потоков данных, действий.
43. Описание классов и их взаимосвязей.
44. Динамика поведения объектов, диаграммы перехода состояний.
45. Диаграммы объектов.
46. Видимость и синхронизация объектов, временные диаграммы.
47. Диаграмма процессов.
48. Обработка исключительных ситуаций.
49. Рабочие продукты, методологии и средства анализа и проектирования.

Тема 9. Технологии коллективной разработки программного обеспечения

50. Обзор и классификация средств поддержки коллективной разработки программногообеспечения.
51. Программные средства планирования и управления процессом разработки.
52. Сетевые графики и диаграммы рабочего процесса.
53. Сценарии выполнения работ, согласование графиков.
54. Применение систем управления документами.
55. Инструментальные средства верификации и тестирования программ.
56. Планирование и автоматизированная генерация тестов.
57. Сценарии тестирования.
58. Анализаторы профиля выполнения теста.
59. Репозиторий тестов.
60. Контроль показателей качества.