|  |  |
| --- | --- |
| Министерство науки и высшего образования Российской Федерации | |
| Федеральное государственное бюджетное образовательное учреждение | |
| высшего образования | |
| «Российский государственный университет им. А.Н. Косыгина | |
| (Технологии. Дизайн. Искусство)» | |
|  | |
| Институт | Мехатроники и информационных технологий |
| Кафедра | Информационных технологий |

|  |  |  |
| --- | --- | --- |
| **РАБОЧАЯ ПРОГРАММА**  **УЧЕБНОЙ ДИСЦИПЛИНЫ** | | |
| **Технологии обработки информации** | | |
| Уровень образования | бакалавриат | |
| Направление подготовки | 09.03.02 | Информационные системы и технологии |
| Профиль | Информационные системы и технологии в индустрии моды | |
| Срок освоения образовательной программы по очной форме обучения | 4 года | |
| Форма обучения | очная | |

|  |  |  |  |
| --- | --- | --- | --- |
| Рабочая программа учебной дисциплины «Технологии обработки информации»основной профессиональной образовательной программы высшего образования*,* рассмотрена и одобрена на заседании кафедры, протокол № 14 от 05.07.2021 г. | | | |
| Разработчик рабочей программы «Технологии обработки информации»: | | | |
|  | канд. техн. наук, доцент | А. А. Семенов | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
| Заведующий кафедрой: | | канд. техн. наук, доцент И. Б. Разин |

# ОБЩИЕ СВЕДЕНИЯ

* + - 1. Учебная дисциплина «Технологии обработки информации» изучается в четвертом семестре.
      2. Курсовая работа/Курсовой проект – не предусмотрены.

## Формы промежуточной аттестации:

|  |  |
| --- | --- |
| четвертый семестр | - зачет с оценкой |
|  |  |

## Место учебной дисциплины в структуре ОПОП

* + - 1. Учебная дисциплина «Технологии обработки информации» относится к обязательной части.
      2. Основой для освоения дисциплины являются результаты обучения по предшествующим дисциплинам и практикам:
    - Введение в профессию;
    - Информатика;
    - Технология программирования.
      1. Результаты обучения по учебной дисциплине, используются при изучении следующих дисциплин:
    - Компьютерная графика;
    - Компьютерное проектирование;
    - Управление данными;
    - Инфокоммуникационные системы и сети.

# ЦЕЛИ И ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

* + - 1. Целями изучения дисциплины «Технологии обработки информации» являются:
* изучение способов представления и структурирования информации о явлениях и процессах в окружающем мире применительно к своей профессиональной деятельности;
* освоение методов ориентирования и взаимодействия с ресурсами информационной среды, осуществления выбора различных моделей использования информационных и коммуникационных технологий в индустрии моды:
* изучение методов построения алгоритмов и основных этапов разработки и создания современных программных продуктов;
* освоение подходов к построению рациональных диалоговых интерфейсов, ориентированных на пользователя;
* изучение базовых правил и принципов современного объектно-ориентированного и визуального программирования;
* изучение методов поиска, анализа и обработки информации;
* формирование у обучающихся компетенций, установленных образовательной программой в соответствии с ФГОС ВО по данной дисциплине.
  + - 1. Результатом обучения по учебной дисциплине является овладение обучающимися знаниями, умениями, навыками и опытом деятельности, характеризующими процесс формирования компетенций и обеспечивающими достижение планируемых результатов освоения учебной дисциплины.

## Формируемые компетенции, индикаторы достижения компетенций, соотнесённые с планируемыми результатами обучения по дисциплине:

| **Код и наименование компетенции** | **Код и наименование индикатора**  **достижения компетенции** | **Планируемые результаты обучения**  **по дисциплине** |
| --- | --- | --- |
| ОПК-3  Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности | ИД-ОПК-3.1  Владение методами поиска и анализа информации для подготовки документов на основе информационной и библиографической культуры с применением  информационно-коммуникационных технологий, с учетом соблюдения авторского права и требований информационной безопасности | - Оценивает сущность и значение информационных технологий в развитии современного общества; владеет основными методами, способами и средствами получения, хранения и обработки информации;  - Использует навыки работы с программными средствами для обработки информации с соблюдением основных требований информационной безопасности.  - Применяет методики использования программных средств для решения практических задач в профессиональной деятельности;  - Умеет разрабатывать современные эффективные интерфейсы «человек - электронно-вычислительная машина»;  - Рационально оценивает и обосновывает принимаемые проектные решения, составляет блок-схемы алгоритмов и реализует их на языках программирования высокого уровня;  - Осуществляет практическую реализацию информационно-коммуникационных программ и компьютерных приложений, выполняет тестовые примеры для проверки их корректности и эффективности. |
| ИД-ОПК-3.2  Подготовка библиографических списков по учебным и научно-исследовательским работам |

# СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

* + - 1. Общая трудоёмкость учебной дисциплины по учебному плану составляет:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| по очной форме обучения – | 3 | **з.е.** | 108 | **час.** |

## Структура учебной дисциплины для обучающихся по видам занятий (очная форма обучения)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Структура и объем дисциплины** | | | | | | | | | |
| **Объем дисциплины по семестрам** | **форма промежуточной аттестации** | **всего, час** | **Контактная аудиторная работа, час** | | | | **Самостоятельная работа обучающегося, час** | | |
| **лекции, час** | **практические занятия, час** | **лабораторные занятия, час** | **практическая подготовка, час** | ***курсовая работа/***  ***курсовой проект*** | **самостоятельная работа обучающегося, час** | **промежуточная аттестация, час** |
| 4 семестр | зачет с оценкой | 108 | 36 |  | 36 | 10 |  | 36 |  |
| Всего: |  | 108 | 36 |  | 36 | 10 |  | 36 |  |

## Структура учебной дисциплины для обучающихся по разделам и темам дисциплины: (очная форма обучения)

| **Планируемые (контролируемые) результаты освоения:**  **код(ы) формируемой(ых) компетенции(й) и индикаторов достижения компетенций** | **Наименование разделов, тем;**  **форма(ы) промежуточной аттестации** | **Виды учебной работы** | | | | **Самостоятельная работа, час** | **Виды и формы контрольных мероприятий, обеспечивающие по совокупности текущий контроль успеваемости;**  **формы промежуточного контроля успеваемости** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Контактная работа** | | | |
| **Лекции, час** | **Практические занятия, час** | ***Лабораторные работы/ индивидуальные занятия, час*** | **Практическая подготовка, час** |
|  | **Четвертыйсеместр** | | | | | | |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел I. Интегрированная среда разработки и технологии программирования** | 4 | х | 3 | 1 | 4 |  |
| Лекция 1.1. Интегрированная среда разработки как инструмент для создания приложений. Технологии программирования для обработки информации | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 1.1. Графика. Структура Figura {}; |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел II. Структуры** | 4 | х | 3 | 1 | 4 |  |
| Лекция 2.1. Структуры в С++ | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 2.1. Сортировки Яндекса |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел III. Динамические структуры данных** | 4 | х | 3 | 1 | 4 |  |
| Лекция 3.1. Динамические структуры данных в С++ | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 3.1. Визуализация сортировки Яндекса |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел IV. Классы** | 6 | х | 6 | 1 | 6 |  |
| Лекция 4.1. Классы в С++ | 6 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 4.1. Классы. Инкапсуляция. Реализация класса MyTime |  |  | 3 |  |  |  |
| Лабораторная работа № 4.2. Классы. Инкапсуляция. Реализация класса PayRoad |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел V. Массивы и классы** | 4 | х | 3 | 1 | 4 |  |
| Лекция 5.1. Массивы и классы в С++ | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 5.1. Классы. Операторные функции. Реализация класса MyInt для перегрузки операций |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел VI. Перегрузка операций** | 4 | х | 6 | 2 | 4 |  |
| Лекция 6.1. Перегрузка операций | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 6.1. Классы. Приведение типов. Реализация класса Stroka |  |  | 3 | 1 |  |  |
| Лабораторная работа № 6.2. Классы. Реализация класса авиарейсов |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел VII. Наследование** | 6 | х | 9 | 2 | 6 |  |
| Лекция 7.1. Наследование в С++ | 6 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 7.1. Классы. Наследование |  |  | 3 |  |  |  |
| Лабораторная работа № 7.2. Классы. Наследование при изначальной разработке программы |  |  | 3 | 1 |  |  |
| Лабораторная работа № 7.3. Классы. Наследование и отработка взаимодействия с компонентом StringGrid |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
| ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 | **Раздел VIII. Виртуальные и дружественные функции** | 4 | х | 3 | 1 | 4 |  |
| Лекция 8.1. Виртуальные и дружественные функции в С++ | 4 |  |  |  |  | Контроль посещаемости. |
| Лабораторная работа № 8.1. Классы. Наследование и виртуальные методы |  |  | 3 | 1 |  | Выполнение лабораторной работы. |
|  | Зачет с оценкой | х | х | х | х | х | Электронное тестирование. |
|  | **ИТОГО за четвертый семестр** | **36** |  | **36** | **10** | **36** | **Зачет с оценкой** |
|  | **ИТОГО за весь период** | **36** |  | **36** | **10** | **36** |  |

## Краткое содержание учебной дисциплины

|  |  |  |
| --- | --- | --- |
| **№ пп** | **Наименование раздела и темы дисциплины** | **Содержание раздела (темы)** |
| **Четвертый семестр** | | |
| **Раздел I** | **Интегрированная среда разработки и технологии программирования** | |
| Лекция 1.1 | Интегрированная среда разработки как инструмент для создания приложений. Технологии программирования для обработки информации. | Введение. Интегрированная среда разработки как инструмент для создания приложений. Характеристика основных технологий программирования. Структуры данных. Правила кодирования, документирования и основные этапы создания программного обеспечения. |
| Лабораторная работа № 1.1 | Графика. Структура Figura {}; | Лабораторная работа посвящена изучению взаимодействия с геометрическими примитивами и отработке структур C++. В рамках данной работы реализуется визуальное приложение согласно представленному интерфейсу. |
| **Раздел II** | **Структуры** | |
| Лекция 2.1 | Структуры в С++. | Структуры, структуры и функции, массивы структур, поиск в массиве структур, вложенность структур, рекурсия, алгоритм быстрой сортировки, массивы структур и бинарные файлы. |
| Лабораторная работа № 2.1 | Сортировки Яндекса. | Лабораторная работа посвящена изучению основных алгоритмов сортировок и способов их реализации на примере Яндекса (см. https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii). Лабораторную работу необходимо реализовать в виде визуального приложения. Нужно изучить представленный иллюстрированный пример реализации на C++ основных видов сортировок. Необходимо воспроизвести реализацию проекта и по аналогии доделать пирамидальную сортировку и сортировку слиянием. Выполнить оптимизацию кода и абстракцию. |
| **Раздел III** | **Динамические структуры данных** | |
| Лекция 3.1 | Динамические структуры данных в С++ | Понятие и предназначение динамических структур данных (ДСД). Характеристика и синтаксис таких ДСД, как линейные списки, стеки, очереди и бинарные деревья.  Возможные области применения и операции над ДСД. |
| Лабораторная работа № 3.1 | Визуализация сортировки Яндекса | Лабораторная работа посвящена визуализации алгоритмов сортировки Яндекса (см. https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii). Лабораторную работу необходимо выполнить в виде визуального приложения. Разрабатывается визуальное приложение, позволяющее визуализировать процесс сортировки целочисленного массива из 12 случайных элементов с анимацией. Допускается использовать подходящий стандартный компонент, либо конструкцию из прямоугольников. Визуализируемую сортировку пользователь выбирает в списке (компонент ComboBox). Замедление просто реализовать с помощью связки sleep() + ProcessMessages(). |
| **Раздел IV** | **Классы** | |
| Лекция 4.1 | Классы в С++ | Основные свойства ООП (инкапсуляция, наследование и полиморфизм). Понятие и элементы класса. Отличие структур от классов. Синтаксис описания класса.  Спецификаторы доступа. Методы класса, отличие методов от функций. Доступ к методам класса. Конструкторы. Определение методов класса вне класса. Методы, возвращающие значения. Классы и память.  Статические данные класса. Константные методы. Деструкторы. |
| Лабораторная работа № 4.1 | Классы. Инкапсуляция. Реализация класса MyTime | Лабораторная работа посвящена изучению классов в ООП на C++. Требуется:  1) Изучить и повторить представленный иллюстрированный пример, демонстрирующий реализацию следующей задачи. Разработать визуальное приложение, в котором необходимо создать класс с именем MyTime, содержащий три поля типа int, предназначенные для хранения часов, минут и секунд. Один из конструкторов класса должен инициализировать поля нулевыми значениями, а другой конструктор – заданным набором значений. Создайте метод класса, который будет выводить значения полей на экран в формате 23:59:59, и метод, складывающий значения двух объектов типа MyTime, передаваемых в качестве аргументов. В обработчике события ButtonClick следует создать два инициализированных объекта и один неинициализированный объект, затем сложить два инициализированных значения, а результат присвоить третьему объекту и вывести его значение на экран (например, 13:23:50 + 10:52:50 = 0:16:40).  2) Повторив рассмотренный в п.1 пример, необходимо обеспечить возможность ввода пользователем значений переменных Т1 и Т2. Для этого необходимо дополнительно разместить на форме два компонента LabeledEdit, которые и будут обеспечивать ввод значений для Т1 и Т2, и реализовать считывание этих значений в обработчике Button1Click. |
| Лабораторная работа № 4.2 | Классы. Инкапсуляция. Реализация класса PayRoad | Лабораторная работа посвящена изучению классов в ООП на C++. Необходимо изучить и повторить представленный иллюстрированный пример, демонстрирующий реализацию класса платной дороги с тарифом 500,50 за проезд. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем PayRoad, содержащий три поля:  int Cars;  float Cash; // наличные  float NonCash; // безналичные  Они предназначены для хранения кол-ва машин, наличных и безналичных платежей. Нулевой конструктор класса должен инициализировать поля нулевыми значениями. Необходимо создать четыре метода класса, которые будут увеличивать счетчик проехавших машин, выводить в LabeledEdit кол-во проехавших машин, выводить суммы платежей (наличных и безналичных), считать и выводить сводные данные. |
| **Раздел V** | **Массивы и классы** | |
| Лекция 5.1 | Массивы и классы в С++ | Массивы и классы. Обработка массивов объектов. Совмещение объектно-ориентированной технологии и процедурной. Строковый класс string и его характеристика. Строковый тип AnsiString и его характеристика. |
| Лабораторная работа № 5.1 | Классы. Операторные функции. Реализация класса MyInt для перегрузки операций | Лабораторная работа посвящена изучению классов в ООП на C++. Нужно изучить и повторить проиллюстрированный пример, демонстрирующий реализацию класса MyInt, демонстрирующего перегрузку арифметических операций и операций сравнения для объектов этого класса. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем MyInt, содержащий одно поле:  int I;  В классе нужно реализовать нулевой конструктор, инициализирующий поля нулевыми значениями, и ненулевой, инициализирующий поля класса значениями. Необходимо создать семь методов класса, которые будут реализовывать операции +, -, \*, /, <, == с объектами класса и выводить результат в Memo. В окна LabeledEdit пользователь вводит числа и нажимает кнопки операций, результат выводится в Memo методом класса show(). Повторив рассмотренный пример, необходимо самостоятельно доделать реализацию кнопок -, \*, / (обработать ситуацию деления на ноль), ==. |
| **Раздел VI** | **Перегрузка операций** | |
| Лекция 6.1 | Перегрузка операций | Синтаксис и принцип работы операторных функций. Правила перегрузки. Перегрузка унарных, арифметических операций, операций сравнения. Перегрузка операции приведения типа. Преобразование объектов в основные типы и наоборот. Преобразование объектов одного класса в объекты другого класса. |
| Лабораторная работа № 6.1 | Классы. Приведение типов. Реализация класса Stroka | Лабораторная работа посвящена изучению классов в ООП на C++. Необходимо изучить и повторить проиллюстрированный пример, демонстрирующий реализацию следующей задачи. На основе типа char создайте класс Stroka. Перегрузите операцию приведения строки типа char к типу Stroka и наоборот. Напишите визуальное приложение (Windows VCL Application) для проверки этого класса. |
| Лабораторная работа № 6.2 | Классы. Реализация класса авиарейсов | Лабораторная работа посвящена изучению классов в ООП на C++. Необходимо изучить и повторить проиллюстрированный пример, демонстрирующий реализацию класса Reis, который положен в основу реализации списка записей авиарейсов. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем Reis, содержащий поля:  char name[N];  char type[N];  int kol;  float price;  В приложении реализовываются возможности добавления записей в таблицу StringGrid, записи их в текстовый файл, считывания оттуда и сортировки рейсов. Здесь в окна LabeledEdit пользователь вводит наименование рейса, тип самолета, кол-во билетов, цену одного билета и нажимает кнопки операций. Такие компоненты, как SaveDialog и OpenDialog настройте как для текстовых файлов (DefaultExt: txt; Filter: Текстовые файлы \*.txt). Остальные компоненты можно особо не настраивать и оставить настройки по умолчанию. Для StringGrid изменение настроек делается программно в коде. Повторив рассмотренный пример, необходимо самостоятельно исправить самую основную недоработку этой программы. Дело в том, что приложение не оптимизировано для ввода наименований рейсов и типов самолетов с пробелами. Если вводить их с пробелами, то процедуры считывания из файла и сортировки работают некорректно. Также в программе имеется ошибка, которая не позволяет считать последнюю строку из файла и вывести её в StringGrid. Эту ошибку надо тоже исправить. |
| **Раздел VII** | **Наследование** | |
| Лекция 7.1 | Наследование в С++ | Понятие и предназначение наследования. Синтаксис наследования. Спецификаторы доступа при наследовании. Конструкторы производных классов. Перегрузка функций.  Применение наследования при первоначальной разработке объектно-ориентированной программы. Иерархия классов. Общее и частное наследование, комбинации доступа.  Множественное наследование. Включение, классы в классах. |
| Лабораторная работа № 7.1 | Классы. Наследование | Лабораторная работа посвящена изучению классов в ООП на C++. Требуется изучить и повторить проиллюстрированный пример, демонстрирующий реализацию базового класса Float и производного класса FloatPr. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором на основе стандартного типа float создается базовый класс Float, имеющий два конструктора, метод вывода на экран и метод для перегрузки арифметической операции +. Используя общее наследование, создается производный класс FloatPr, добавляющий возможность использования операций -, \*, /. Далее идет проверка производного класса. |
| Лабораторная работа № 7.2 | Классы. Наследование при изначальной разработке программы | Лабораторная работа посвящена изучению классов в ООП на C++. Требуется изучить и повторить проиллюстрированный пример, демонстрирующий применение технологии наследования при изначальном проектировании приложения. Используя известный по лекциям класс Tovar, в примере создаётся два производных от него класса: 1) TovarProd, добавляющий возможность хранить информацию о сроке хранения и температуре хранения продуктовых товаров; 2) TovarProm, позволяющий хранить информацию в соответствии с полями базового класса. Созданное визуальное приложение (Windows VCL Application) должно позволять: вводить информацию либо о продуктовых товарах, либо о промышленных товарах; выводить общую стоимость товаров, имеющихся на складе. Поставленную задачу удобно решать с применением технологии наследования при изначальной разработке приложения. По условию задачи, у нас две категории товаров, у которых первые три поля (наименование, номер, цена) совпадают. И действие по добавлению записи тоже совпадает. Таким образом, создадим базовый класс Tovar и два производных от него класса для продуктовых товаров и для промышленных. Производных класс TovarProd для продуктовых товаров будет задействовать все поля и методы базового класса и добавлять еще некоторые специфические именно для себя. Производный класс TovarProm вообще не требует никаких доработок, так как ему полностью хватает возможностей базового класса. Далее создадим массивы для работы с классами продуктовых и промышленных товаров. После создадим обработчик события нажатия на кнопку Button1 и обработчик события нажатия на кнопку Button2, прописав в них логику добавления товаров в соответствующие массивы и вывода записей в Memo1. В обработчик события нажатия кнопки Button3 запишем логику подсчета общей стоимости добавленных товаров. |
| Лабораторная работа № 7.3 | Классы. Наследование и отработка взаимодействия с компонентом StringGrid | Лабораторная работа посвящена изучению классов в C++. Необходимо изучить и повторить проиллюстрированный пример, демонстрирующий применение технологии наследования при изначальном проектировании приложения. Взяв за основу проект из предыдущей лабораторной работы, мы создадим его улучшенную версию, которая позволит добавлять товары не в компонент Memo, а в компонент StringGrid. Также реализуем сохранение данных в текстовый файл и считывание их из него в компонент StringGrid. Поставленную задачу будем решать с применением технологии наследования при изначальной разработке приложения. Создадим базовый класс и распишем производные классы. Создадим массивы для работы с классами продуктовых и промышленных товаров. После создадим обработчик события нажатия на кнопку Button1 и обработчик события нажатия на кнопку Button2, прописав в них логику добавления товаров в соответствующие массивы и вывода записей в StirngGrid1. В обработчик события нажатия кнопки Button3 запишем логику подсчета общей стоимости добавленных товаров. Далее, создадим обработчик события OnCreate для формы и обработчики нажатия кнопок для сохранения данных в текстовый файл и их загрузки из файла. В последней процедуре Button5Click, создается объект SL, у которого тип данных - StringList. Это строковый список. Удобная конструкция для построчного считывания из файла и заполнения строк таблицы StrinGrid этими данными. Реализовав рассмотренный пример, необходимо добавить возможность правильной работы кнопки "Общая стоимость", если данные не добавлялись вручную, а были загружены из файла. Сейчас же, если запустить программу и загрузить данные из файла кнопкой "Загрузить", то программа выдаст сообщение "Нет товаров!". Это верно, так как при заполнении табличной части компонента StringGrid, мы не записываем данные в массивы товаров, соответственно массивы остаются пустыми. Это надо исправить. Дополните процедуру Button5Click и создайте дополнительный метод класса, позволяющий добавлять данные из StringGrid. То есть, можно сделать по аналогии с методом, который добавляет записи из эдитов:  void dobav\_zap() { // метод для добавления записей  if(VidTovara==0) {  strcpy(name, AnsiString(Form1->LabeledEdit1->Text).c\_str());  number = StrToInt(Form1->LabeledEdit2->Text);  price = StrToFloat(Form1->LabeledEdit3->Text);  }  if(VidTovara==1) {  strcpy(name, AnsiString(Form1->LabeledEdit6->Text).c\_str());  number = StrToInt(Form1->LabeledEdit7->Text);  price = StrToFloat(Form1->LabeledEdit8->Text);  }  }    Необходимо разработать метод, в котором они добавлялись бы из StringGrid. |
| **Раздел VIII** | **Виртуальные и дружественные функции** | |
| Лекция 8.1 | Виртуальные и дружественные функции в С++ | Предназначение и синтаксис виртуальных функций. Указатели на базовый класс. Наследование и массивы указателей на базовый класс, применение таких массивов. Абстрактные классы и чистые виртуальные функции. Виртуальные деструкторы. Виртуальные базовые классы и устранение неоднозначности при множественном наследовании. Предназначение и синтаксис дружественных функций. Применение процедурной технологии создания программ в объектно-ориентированной программе. Дружественные классы. Указатель this. |
| Лабораторная работа № 8.1 | Классы. Наследование и виртуальные методы | Лабораторная работа посвящена изучению классов в ООП на C++. Необходимо изучить и повторить пример, демонстрирующий применение технологии наследования с виртуальными методами. Взяв за основу проект из предыдущей лабораторной работы, мы создадим его улучшенную версию, которая позволит работать не с двумя массивами, а с одним универсальным, который может хранить данные как продуктовых товаров, так и промышленных. Также реализуем сохранение данных в текстовый файл и считывание их из него в компонент StringGrid (по аналогии с предыдущей л/р). Также реализуем нормальное функционирование кнопки "Общая стоимость", которая сможет рассчитывать общую стоимость как добавленных вручную товаров, так и загруженных из файла. Реализовав рассмотренный пример, необходимо добавить возможности: а) сортировки товаров по наименованию и цене; б) поиска (по всем свойствам товаров). |

## Организация самостоятельной работы обучающихся

Самостоятельная работа студента – обязательная часть образовательного процесса, направленная на развитие готовности к профессиональному и личностному самообразованию, на проектирование дальнейшего образовательного маршрута и профессиональной карьеры.

Самостоятельная работа обучающихся по дисциплине организована как совокупность аудиторных и внеаудиторных занятий и работ, обеспечивающих успешное освоение дисциплины.

Аудиторная самостоятельная работа обучающихся по дисциплине выполняется на учебных занятиях под руководством преподавателя и по его заданию*.* Аудиторная самостоятельная работа обучающихся входит в общий объем времени, отведенного учебным планом на аудиторную работу, и регламентируется расписанием учебных занятий.

Внеаудиторная самостоятельная работа обучающихся – планируемая учебная, научно-исследовательская, практическая работа обучающихся, выполняемая во внеаудиторное время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия, расписанием учебных занятий не регламентируется.

Внеаудиторная самостоятельная работа обучающихся включает в себя:

подготовку к лекциям, практическим занятиям, лабораторным работам и экзамену;

изучение специальной рекомендованной литературы;

изучение разделов/тем, не выносимых на лекции и практические занятия самостоятельно;

подготовка к выполнению лабораторных работ;

подготовка к практическим занятиям*;*

подготовка к компьютерному тестированию на промежуточных аттестациях;

выполнение индивидуальных заданий;

подготовка к промежуточной аттестации в течение семестра.

Самостоятельная работа обучающихся с участием преподавателя в форме иной контактной работы предусматривает групповую и (или) индивидуальную работу с обучающимися и включает в себя:

проведение индивидуальных и групповых консультаций по отдельным темам/разделам дисциплины;

проведение консультаций перед экзаменом, перед зачетом с оценкой;

консультации по организации самостоятельного изучения отдельных разделов/тем, базовых понятий учебных дисциплин профильного/родственного бакалавриата, которые формировали ОПК и ПК, в целях обеспечения преемственности образования.

Перечень разделов/тем, полностью или частично отнесенных на самостоятельное изучение с последующим контролем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ пп** | **Наименование раздела /темы *дисциплины,* выносимые на самостоятельное изучение** | **Задания для самостоятельной работы** | **Виды и формы контрольных мероприятий**  **(учитываются при проведении текущего контроля)** | **Трудоемкость, час** |
| **Раздел I** | **Интегрированная среда разработки и технологии программирования** | | | |
| Лабораторная работа № 1.1 | Графика. Структура Figura {}; | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 4 |
| **Раздел II** | Структуры | | | |
| Лабораторная работа № 2.1 | Сортировки Яндекса | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 4 |
| **Раздел III** | **Динамические структуры данных** | | | |
| Лабораторная работа № 3.1 | Визуализация сортировки Яндекса | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 4 |
| **Раздел IV** | **Классы** | | | |
| Лабораторная работа № 4.1 | Классы. Инкапсуляция. Реализация класса MyTime. | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 3 |
| Лабораторная работа № 4.2 | Классы. Инкапсуляция. Реализация класса PayRoad | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 3 |
| **Раздел V** | **Массивы и классы** | | | |
| Лабораторная работа № 5.1 | Классы. Операторные функции. Реализация класса MyInt для перегрузки операций | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 4 |
| **Раздел VI** | **Перегрузка операций** | | | |
| Лабораторная работа № 6.1 | Классы. Приведение типов. Реализация класса Stroka | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 2 |
| Лабораторная работа № 6.2 | Классы. Реализация класса авиарейсов | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 2 |
| **Раздел VII** | **Наследование** | | | |
| Лабораторная работа № 7.1 | Классы. Наследование | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 2 |
| Лабораторная работа № 7.2 | Классы. Наследование при изначальной разработке программы | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 2 |
| Лабораторная работа № 7.3 | Классы. Наследование и отработка взаимодействия с компонентом StringGrid | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 2 |
| **Раздел VIII** | **Виртуальные и дружественные функции** | | | |
| Лабораторная работа № 8.1 | Классы. Наследование и виртуальные методы | Изучение научной и технической литературы, нормативных документов, стандартов языков программирования. Работа с материалами конспекта лекций. Анализ задания к лабораторной работе, выбор способов её выполнения. Осваивание методов объектно-ориентированного и визуального программирования. Изучение элементов системы разработки программ и операторов языка для выполнения задания лабораторной работы. | Выполнение лабораторной работы. | 4 |

## Применение электронного обучения, дистанционных образовательных технологий

При реализации программы учебной дисциплины возможно применение электронного обучения и дистанционных образовательных технологий.

Реализация программы учебной дисциплины с применением электронного обучения и дистанционных образовательных технологий регламентируется действующими локальными актами университета.

Применяются следующие разновидности реализации программы с использованием ЭО и ДОТ.

В электронную образовательную среду, по необходимости, могут быть перенесены отдельные виды учебной деятельности:

|  |  |  |  |
| --- | --- | --- | --- |
| **использование**  **ЭО и ДОТ** | **использование ЭО и ДОТ** | **объем, час** | **включение в учебный процесс** |
| смешанное обучение | лекции | 36 | в соответствии с расписанием учебных занятий |
| лабораторные занятия | 36 |

# РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, КРИТЕРИИ ОЦЕНКИ УРОВНЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ, СИСТЕМА И ШКАЛА ОЦЕНИВАНИЯ

## Соотнесение планируемых результатов обучения с уровнями сформированности компетенций.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Уровни сформированности компетенции(-й)** | **Итоговое количество баллов**  **в 100-балльной системе**  **по результатам текущей и промежуточной аттестации** | **Оценка в пятибалльной системе**  **по результатам текущей и промежуточной аттестации** | **Показатели уровня сформированности** | | |
| **универсальной(-ых)**  **компетенции(-й)** | **общепрофессиональной(-ых) компетенций** | **профессиональной(-ых)**  **компетенции(-й)** |
|  | ОПК-3:  ИД-ОПК-3.1  ИД-ОПК-3.2 |  |
| высокий |  | отлично/  зачтено (отлично)/  зачтено |  | Обучающийся:   * исчерпывающе и логически стройно излагает учебный материал, умеет связывать теорию с практикой, справляется с решением задач профессиональной направленности высокого уровня сложности, правильно обосновывает принятые решения; * способен уверенно использовать современные системы разработки прикладных программ с эффективными графическими интерфейсами и системы коммуникации в сети Internet; * показывает творческие способности в понимании и практическом использовании языков высокого уровня, использовании визуальных компонентов разработки приложений графических интерфейсов; * дополняет теоретическую информацию сведениями, самостоятельно полученными из источников научно-технической информации; * способен провести целостный анализ среды разработки современных программ на основе объектно-ориентированного и визуального программирования; * свободно ориентируется в учебной и профессиональной литературе; * дает развернутые, исчерпывающие, профессионально грамотные ответы на вопросы, в том числе, дополнительные. |  |
| повышенный |  | хорошо/  зачтено (хорошо)/  зачтено |  | Обучающийся:   * достаточно подробно, грамотно и по существу излагает изученный материал, приводит и раскрывает в тезисной форме основные понятия; * анализирует современные Технология программирования с незначительными пробелами; * способен использовать только основные функциональные возможности систем разработки программ и систем коммуникации в сети Internet; * способен провести анализ основных элементов разработки современных программ на основе объектно-ориентированного и визуального программирования; * допускает единичные негрубые ошибки; * достаточно хорошо ориентируется в учебной и профессиональной литературе; * ответ отражает знание теоретического и практического материала, не допуская существенных неточностей. |  |
| базовый |  | удовлетворительно/  зачтено (удовлетворительно)/  зачтено |  | Обучающийся:   * демонстрирует теоретические знания основного учебного материала дисциплины в объеме, необходимом для дальнейшего освоения ОПОП; * с неточностями излагает принципы и методы разработки современных программ на основе объектно-ориентированного и визуального программирования; * способен использовать отдельные элементы визуальной разработки прикладных программ; * анализирует современные технологии программирования с неточностями и ошибками; * демонстрирует фрагментарные знания основной учебной литературы по дисциплине; * ответ отражает знания на базовом уровне теоретического и практического материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профилю обучения. |  |
| низкий |  | неудовлетворительно/  не зачтено | Обучающийся:   * демонстрирует фрагментарные знания теоретического и практического материал, допускает грубые ошибки при его изложении на занятиях и в ходе промежуточной аттестации; * испытывает серьёзные затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, не владеет необходимыми для этого навыками и приёмами; * не способен проанализировать учебно-методическую, техническую и научную литературу; * не владеет основными принципами и навыками работы в современных средах разработки прикладных программ, не умеет пользоваться системами коммуникации (Internet); * выполняет задания только по образцу и под руководством преподавателя; * ответ отражает отсутствие знаний на базовом уровне теоретического и практического материала в объеме, необходимом для дальнейшей учебы. | | |

# ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ, ВКЛЮЧАЯ САМОСТОЯТЕЛЬНУЮ РАБОТУ ОБУЧАЮЩИХСЯ

* + - 1. При проведении контроля самостоятельной работы обучающихся, текущего контроля и промежуточной аттестации по учебной дисциплине «Технологии обработки информации»проверяется уровень сформированности у обучающихся компетенций и запланированных результатов обучения по дисциплине*,* указанных в разделе 2 настоящей программы.

## Формы текущего контроля успеваемости, примеры типовых заданий:

| **№ пп** | **Формы текущего контроля** | * + - 1. **Примеры типовых заданий** |
| --- | --- | --- |
| Лабораторная работа № 1.1 | Выполнение лабораторной работы. | Графика. Структура Figura {};.  Лабораторная работа посвящена изучению взаимодействия с геометрическими примитивами и отработке структур C++. В рамках данной работы реализуется визуальное приложение согласно представленному интерфейсу. |
| Лабораторная работа № 2.1 | Выполнение лабораторной работы. | Сортировки Яндекса.  Лабораторная работа посвящена изучению основных алгоритмов сортировок и способов их реализации на примере Яндекса (см. https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii). Лабораторную работу необходимо реализовать в виде визуального приложения. Нужно изучить представленный иллюстрированный пример реализации на C++ основных видов сортировок.  Пример реализации на C++ основных видов сортировок:  1) Кнопка 1 формирует случайным образом массив целых чисел из 20000 элементов и выводит, например, в MEMO1.  2) НА ФОРМЕ ИМЕЕТСЯ радиогруп (8 радиоточек для каждой из сортировок).  3) Кнопка 2 - сортирует массив случайных чисел (вызывается функция для выбранной пользователем сортировки).  4) Выводим отсортированный массив и время в миллисекундах, затраченное в ходе сортировки.  Интерфейс приложения:    Необходимо воспроизвести реализацию проекта и по аналогии доделать пирамидальную сортировку и сортировку слиянием. Выполнить оптимизацию кода (для ускорения операций) и абстракцию. |
| Лабораторная работа № 3.1 | Выполнение лабораторной работы. | Визуализация сортировки Яндекса.  Лабораторная работа посвящена визуализации алгоритмов сортировки Яндекса (см. https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii). Лабораторную работу необходимо выполнить в виде визуального приложения. Разрабатывается визуальное приложение, позволяющее визуализировать процесс сортировки целочисленного массива из 12 случайных элементов с анимацией. Допускается использовать подходящий стандартный компонент, либо конструкцию из прямоугольников. Визуализируемую сортировку пользователь выбирает в списке (компонент ComboBox). Замедление просто реализовать с помощью связки sleep() + ProcessMessages(). |
| Лабораторная работа № 4.1 | Выполнение лабораторной работы. | Классы. Инкапсуляция. Реализация класса MyTime.  Лабораторная работа посвящена изучению классов в ООП на C++.  1) Изучить и повторить самостоятельно иллюстрированный пример (см. ниже), демонстрирующий реализацию следующей задачи. Разработать визуальное приложение, в котором необходимо создать класс с именем MyTime, содержащий три поля типа int, предназначенные для хранения часов, минут и секунд. Один из конструкторов класса должен инициализировать поля нулевыми значениями, а другой конструктор – заданным набором значений. Создайте метод класса, который будет выводить значения полей на экран в формате 23:59:59, и метод, складывающий значения двух объектов типа MyTime, передаваемых в качестве аргументов. В обработчике события ButtonClick следует создать два инициализированных объекта и один неинициализированный объект, затем сложить два инициализированных значения, а результат присвоить третьему объекту и вывести его значение на экран (например, 13:23:50 + 10:52:50 = 0:16:40).  Иллюстрированный пример разработки приложения  Интерфейс приложения    Исходный код программы  .h-файл:  //---------------------------------------------------------------------------  #ifndef Unit1\_mytimeH  #define Unit1\_mytimeH  //---------------------------------------------------------------------------  #include <System.Classes.hpp>  #include <Vcl.Controls.hpp>  #include <Vcl.StdCtrls.hpp>  #include <Vcl.Forms.hpp>  //---------------------------------------------------------------------------  class TForm1 : public TForm  {  \_\_published: // IDE-managed Components  TMemo \*Memo1;  TButton \*Button1;  void \_\_fastcall Button1Click(TObject \*Sender);  private: // User declarations  public: // User declarations  \_\_fastcall TForm1(TComponent\* Owner);  };  class MyTime {  private:  int chas;  int min;  int sec;  public:  MyTime() { chas=0; min=0; sec=0; }  MyTime(int ch, int m, int s) {  chas=ch; min=m; sec=s;  }  void show();  void summa(MyTime t1, MyTime t2);  };  //---------------------------------------------------------------------------  extern PACKAGE TForm1 \*Form1;  //---------------------------------------------------------------------------  #endif  .cpp-файл:  //---------------------------------------------------------------------------  #include <vcl.h>  #pragma hdrstop  #include "Unit1\_mytime.h"  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  void MyTime::show(){  AnsiString s;  s = IntToStr(chas) + ":" + IntToStr(min) + ":" + IntToStr(sec);  Form1->Memo1->Lines->Add(s);  }  void MyTime::summa(MyTime t1, MyTime t2){  sec = t1.sec + t2.sec;  min = t1.min + t2.min;  chas = t1.chas + t2.chas;  if(sec>=60) { min++; sec-=60; }  if(min>=60) { chas++; min-=60; }  if(chas>=24) chas = chas-24;  }  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  {  Memo1->Clear();  MyTime T1(13,23,50), T2(10,52,50), T3;  T1.show(); T2.show();  T3.summa(T1, T2);  T3.show();  }  //---------------------------------------------------------------------------  2) Повторив рассмотренный в п.1 пример, необходимо обеспечить возможность ввода пользователем значений переменных Т1 и Т2. Для этого необходимо дополнительно разместить на форме два компонента LabeledEdit, которые и будут обеспечивать ввод значений для Т1 и Т2, и реализовать считывание этих значений в обработчике Button1Click. |
| Лабораторная работа № 4.2 | Выполнение лабораторной работы. | Классы. Инкапсуляция. Реализация класса PayRoad.  Лабораторная работа посвящена изучению классов в ООП на C++. Изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию класса платной дороги с тарифом 500,50 за проезд. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем PayRoad, содержащий три поля:  int Cars;  float Cash; // наличные  float NonCash; // безналичные  Они предназначены для хранения кол-ва машин, наличных и безналичных платежей. Нулевой конструктор класса должен инициализировать поля нулевыми значениями. Необходимо создать четыре метода класса, которые будут увеличивать счетчик проехавших машин, выводить в LabeledEdit кол-во проехавших машин, выводить суммы платежей (наличных и безналичных), считать и выводить сводные данные.  Иллюстрированный пример разработки приложения  Интерфейс приложения    Здесь в окна LabeledEdit ничего с клавиатуры вводить нельзя, они отображают просто счетчики проехавших машин, которые увеличиваются после нажатия кнопок Оплата.  Исходный код программы  .h-файл:  //---------------------------------------------------------------------------  #ifndef Unit1\_payroadH  #define Unit1\_payroadH  //---------------------------------------------------------------------------  #include <System.Classes.hpp>  #include <Vcl.Controls.hpp>  #include <Vcl.StdCtrls.hpp>  #include <Vcl.Forms.hpp>  #include <Vcl.ExtCtrls.hpp>  //---------------------------------------------------------------------------  class TForm1 : public TForm  {  \_\_published: // IDE-managed Components  TGroupBox \*GroupBox1;  TLabeledEdit \*LabeledEdit1;  TButton \*Button1;  TGroupBox \*GroupBox2;  TLabeledEdit \*LabeledEdit2;  TButton \*Button2;  TButton \*Button3;  TButton \*Button4;  TButton \*Button5;  void \_\_fastcall Button1Click(TObject \*Sender);  void \_\_fastcall Button3Click(TObject \*Sender);  void \_\_fastcall Button2Click(TObject \*Sender);  void \_\_fastcall Button4Click(TObject \*Sender);  void \_\_fastcall Button5Click(TObject \*Sender);  private: // User declarations  public: // User declarations  \_\_fastcall TForm1(TComponent\* Owner);  };  class PayRoad { // класс платной дороги  private:  int Cars;  float Cash;  float NonCash;  public:  PayRoad() { Cars=0; Cash=0.0; NonCash=0.0; }  void paying(int flag); // увеличиваем счетчик проехавших  void sum(int flag); // суммируем платежи  void total(PayRoad car1, PayRoad car2); // считаем итого по нал. и безнал. платежам  void show(int flag); // выводим showmessage  };  //---------------------------------------------------------------------------  extern PACKAGE TForm1 \*Form1;  //---------------------------------------------------------------------------  #endif  .cpp-файл:  //---------------------------------------------------------------------------  #include <vcl.h>  #pragma hdrstop  #include "Unit1\_payroad.h"  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  PayRoad CarCash, CarNonCash, TotalCars;  void PayRoad::paying(int flag){ // увеличиваем счетчик проехавших  Cars++;  if(flag==1) Cash+=500.50;  if(flag==2) NonCash+=500.50;  }  void PayRoad::show(int flag){  if(flag==1) Form1->LabeledEdit1->Text = Cars;  if(flag==2) Form1->LabeledEdit2->Text = Cars;  }  void PayRoad::sum(int flag) {  if(flag==1) ShowMessage(Cash);  if(flag==2) ShowMessage(NonCash);  }  void PayRoad::total(PayRoad car1, PayRoad car2) {  Cars = car1.Cars + car2.Cars;  Cash = car1.Cash + car2.Cash;  NonCash = car1.NonCash + car2.NonCash;  AnsiString s = "Всего проехало машин: " + IntToStr(Cars) + "." + " Общая выручка: " + FloatToStr(Cash+NonCash);  ShowMessage(s);  }  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  {  CarCash.paying(1);  CarCash.show(1);  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button3Click(TObject \*Sender)  {  CarCash.sum(1);  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button2Click(TObject \*Sender)  {  CarNonCash.paying(2);  CarNonCash.show(2);  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button4Click(TObject \*Sender)  {  CarNonCash.sum(2);  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button5Click(TObject \*Sender)  {  TotalCars.total(CarCash, CarNonCash);  } |
| Лабораторная работа № 5.1 | Выполнение лабораторной работы. | Классы. Операторные функции. Реализация класса MyInt для перегрузки операций.  Лабораторная работа посвящена изучению классов в ООП на C++. Изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию класса MyInt, демонстрирующего перегрузку арифметических операций и операций сравнения для объектов этого класса. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем MyInt, содержащий одно поле:  int I;  В классе нужно реализовать нулевой конструктор, инициализирующий поля нулевыми значениями, и ненулевой, инициализирующий поля класса значениями. Необходимо создать семь методов класса, которые будут реализовывать операции +, -, \*, /, <, == с объектами класса и выводить результат в Memo.  Иллюстрированный пример разработки приложения  Интерфейс приложения    Здесь в окна LabeledEdit пользователь вводит числа и нажимает кнопки операций, результат выводится в Memo методом класса show().  Исходный код программы  В предыдущих лабораторных работах описание класса мы размещали в .h-файле, а в этой работе разместим для упрощения в .cpp-файле.  .cpp-файл:  //---------------------------------------------------------------------------  #include <vcl.h>  #pragma hdrstop  #include "Unit1\_calc.h"  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  class MyInt {  private:  int I;  public:  MyInt() { I=0; }  MyInt(int i) { I=i; }  void show(){  Form1->Memo1->Lines->Add(I);  }  MyInt operator+(MyInt a2) {  return (I + a2.I);  }  bool operator<(MyInt a2) {  if(I < a2.I) return true;  else return false;  }  };  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  {  MyInt i1, i2, i3;  i1 = StrToInt(LabeledEdit1->Text);  i2 = StrToInt(LabeledEdit2->Text);  i3 = i1 + i2;  i3.show();  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button5Click(TObject \*Sender)  {  MyInt i1, i2, i3;  i1 = StrToInt(LabeledEdit1->Text);  i2 = StrToInt(LabeledEdit2->Text);  if(i1 < i2) Form1->Memo1->Lines->Add("Истина");  else Form1->Memo1->Lines->Add("ЛОЖЬ");  }  //---------------------------------------------------------------------------  Повторив рассмотренный пример, необходимо самостоятельно доделать реализацию кнопок -, \*, / (обработать ситуацию деления на ноль), ==. Выполните абстракцию. |
| Лабораторная работа № 6.1 | Выполнение лабораторной работы. | Классы. Приведение типов. Реализация класса Stroka.  Лабораторная работа посвящена изучению классов в ООП на C++. Изучить и повторить проиллюстрированный в видеозаписи пример, демонстрирующий реализацию следующей задачи. На основе типа char создайте класс Stroka. Перегрузите операцию приведения строки типа char к типу Stroka и наоборот. Напишите визуальное приложение (Windows VCL Application) для проверки этого класса. |
| Лабораторная работа № 6.2 | Выполнение лабораторной работы. | Классы. Реализация класса авиарейсов.  Лабораторная работа посвящена изучению классов в ООП на C++. Изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию класса Reis, который положен в основу реализации списка записей авиарейсов. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем Reis, содержащий поля:  char name[N];  char type[N];  int kol;  float price;  В приложении реализовываются возможности добавления записей в таблицу StringGrid, записи их в текстовый файл, считывания оттуда и сортировки рейсов.  Иллюстрированный пример разработки приложения  Интерфейс приложения    Здесь в окна LabeledEdit пользователь вводит наименование рейса, тип самолета, кол-во билетов, цену одного билета и нажимает кнопки операций. Такие компоненты, как SaveDialog и OpenDialog настройте так, как мы всегда с вами делали для текстовых файлов (DefaultExt: txt; Filter: Текстовые файлы \*.txt). Остальные компоненты можно особо не настраивать и оставить настройки по умолчанию. Для StringGrid изменение настроек делается программно в коде.  Исходный код программы  В .h-файле ничего не прописываем.  .cpp-файл:  //---------------------------------------------------------------------------  #include <vcl.h>  #include <fstream.h>  #pragma hdrstop  #include "Unit1\_lr7.h"  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  int i = 0; // счетчик позиций (строк в таблице, записей в массиве)  const int N = 255; // размер строк с наименованием рейса и типом самолета  class Reis {  private:  char name[N];  char type[N];  int kol;  float price;  public:  void dobav\_zap() { // метод для добавления записи  strcpy(name, AnsiString(Form1->LabeledEdit1->Text).c\_str());  strcpy(type, AnsiString(Form1->LabeledEdit2->Text).c\_str());  kol = StrToInt(Form1->LabeledEdit3->Text);  price = StrToFloat(Form1->LabeledEdit4->Text);  }  void show() { // метод для добавления в таблицу  AnsiString s;  s = AnsiString(name) + " " + AnsiString(type) + " " + IntToStr(kol)+ " " + FloatToStr(price);  Form1->StringGrid1->Cells[0][i] = AnsiString(name);  Form1->StringGrid1->Cells[1][i] = AnsiString(type);  Form1->StringGrid1->Cells[2][i] = kol;  Form1->StringGrid1->Cells[3][i] = price;  }  void clear() { // метод для очистки эдитов  Form1->LabeledEdit1->Clear();  Form1->LabeledEdit2->Clear();  Form1->LabeledEdit3->Clear();  Form1->LabeledEdit4->Clear();  }  } ;  const int K = 100; // кол-во записей в массиве  Reis\* reis\_samolet[K]; // массив класса  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button1Click(TObject \*Sender) // Добавить  {  reis\_samolet[i] = new Reis;  reis\_samolet[i]->dobav\_zap();  reis\_samolet[i]->show();  reis\_samolet[i]->clear();  i++;  Form1->StringGrid1->RowCount++; // добавим строчку в таблицу StringGrid  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::FormCreate(TObject \*Sender) // при создании формы  {  //отключим шапочные строку и столбец таблицы  Form1->StringGrid1->FixedCols = 0;  Form1->StringGrid1->FixedRows = 0;  //кол-во столбцов таблицы сделаем равным кол-ву полей класса  Form1->StringGrid1->ColCount = 4;  //кол-во строк таблицы сделаем равным одной  Form1->StringGrid1->RowCount = 1;  //зададим ширину первых двух столбцов  Form1->StringGrid1->ColWidths[0]=N;  Form1->StringGrid1->ColWidths[1]=N;  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button2Click(TObject \*Sender) // Сохранить  {  if(SaveDialog1->Execute()) {  char \*fn = AnsiString(SaveDialog1->FileName).c\_str();  ofstream AllOrder(fn, ios::app); //создаем поток к файлу  // удобнее, конечно, работать с массивом reis\_samolet, но  // отработаем взаимодействие со StringGrid  for(int e = 0; e < StringGrid1->RowCount; e++) {  StringGrid1->Rows[e]->Delimiter = ' ';  AnsiString s = StringGrid1->Rows[e]->DelimitedText;  char \*st=s.c\_str();  AllOrder << st << endl;  }  AllOrder.close();  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button3Click(TObject \*Sender) // Загрузить  {  if(OpenDialog1->Execute()) {  // из выбранного пользователем файла заполняем данными StringGrid  int e=0;  char \*fn = AnsiString(OpenDialog1->FileName).c\_str();  ifstream fin2(fn);  char name\_f[N];  char type\_f[N];  int kol\_f;  float price\_f;  while(!fin2.eof()) {  fin2 >> name\_f >> type\_f >> kol\_f >> price\_f;  Form1->StringGrid1->Cells[0][e] = AnsiString(name\_f);  Form1->StringGrid1->Cells[1][e] = AnsiString(type\_f);  Form1->StringGrid1->Cells[2][e] = kol\_f;  Form1->StringGrid1->Cells[3][e] = price\_f;  e++;  }  fin2.close();  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button4Click(TObject \*Sender) // Сортировка по рейсу  {  TStringList \*SL = new TStringList;  AnsiString S;  int g=0;  for (int i=0; i < StringGrid1->RowCount; i++){  S="";  //if(StringGrid1->Cells[0][i].IsEmpty()) ShowMessage(StringGrid1->Cells[0][i]);  for (int j=0; j < StringGrid1->ColCount; j++) S += StringGrid1->Cells[j][i] + " ";  //ShowMessage(S.Length());  if(S.Length()!=4) { SL->Add(S); g++; }  }  SL->Sort();  for (int i=0; i < g; i++){  StringGrid1->Rows[i]->DelimitedText = SL->Strings[i];  }  delete SL;  }  //---------------------------------------------------------------------------  Повторив рассмотренный пример, необходимо самостоятельно исправить самую основную недоработку этой программы. Дело в том, что приложение не оптимизировано для ввода наименований рейсов и типов самолетов с пробелами. Если вводить их с пробелами, то процедуры считывания из файла и сортировки работают некорректно.  Также в программе имеется ошибка, которая не позволяет считать последнюю строку из файла и вывести её в StringGrid. Эту ошибку надо тоже исправить. |
| Лабораторная работа № 7.1 | Выполнение лабораторной работы. | Классы. Наследование.  Лабораторная работа посвящена изучению классов в ООП на C++. Изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию базового класса Float и производного класса FloatPr. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором на основе стандартного типа float создается базовый класс Float, имеющий два конструктора, метод вывода на экран и метод для перегрузки арифметической операции +. Используя общее наследование, создается производный класс FloatPr, добавляющий возможность использования операций -, \*, /. Далее идет проверка производного класса.  Иллюстрированный пример разработки приложения  Интерфейс приложения    Исходный код программы  В .h-файле для упрощения ничего не прописываем.  .cpp-файл:  //---------------------------------------------------------------------------  #include <vcl.h>  #pragma hdrstop  #include "Unit1\_lr8.h"  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  class Float { // базовый класс  protected:  float f;  public:  Float() { f=0; }  Float(float d) { f=d; }  void show() { Form1->Panel1->Caption = f; }  Float operator+(Float b) { return (f+b.f); }  };  class FloatPr : public Float { // производный класс (общее наследование)  public:  FloatPr() : Float() {}  FloatPr(float v) : Float(v) {}  FloatPr(Float a) : Float(a) {} // для +  FloatPr operator-(FloatPr b) { return (f-b.f); }  FloatPr operator\*(FloatPr b) { return (f\*b.f); }  FloatPr operator/(FloatPr b) { return (f/b.f); }  };  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  {  FloatPr f1, f2, f3;  f1 = StrToFloat(LabeledEdit1->Text);  f2 = StrToFloat(LabeledEdit2->Text);  f3 = f1+f2; // вызывается метод базового класса operator+ и третий конструктор производного класса  f3.show();  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button2Click(TObject \*Sender)  {  FloatPr f1, f2, f3;  f1 = StrToFloat(LabeledEdit1->Text);  f2 = StrToFloat(LabeledEdit2->Text);  f3 = f1-f2; // вызывается метод производного класса operator-  f3.show();  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button3Click(TObject \*Sender)  {  FloatPr f1, f2, f3;  f1 = StrToFloat(LabeledEdit1->Text);  f2 = StrToFloat(LabeledEdit2->Text);  f3 = f1\*f2; // вызывается метод производного класса operator\*  f3.show();  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button4Click(TObject \*Sender)  {  FloatPr f1, f2, f3;  f1 = StrToFloat(LabeledEdit1->Text);  f2 = StrToFloat(LabeledEdit2->Text);  f3 = f1/f2; // вызывается метод производного класса operator/  f3.show();  }  Выполните абстракцию. |
| Лабораторная работа № 7.2 | Выполнение лабораторной работы. | Классы. Наследование при изначальной разработке программы.  Лабораторная работа посвящена изучению классов в ООП на C++. Требуется изучить и повторить проиллюстрированный пример, демонстрирующий применение технологии наследования при изначальном проектировании приложения. Используя известный по лекциям класс Tovar, в примере создаётся два производных от него класса: 1) TovarProd, добавляющий возможность хранить информацию о сроке хранения и температуре хранения продуктовых товаров; 2) TovarProm, позволяющий хранить информацию в соответствии с полями базового класса. Созданное визуальное приложение (Windows VCL Application) должно позволять: вводить информацию либо о продуктовых товарах, либо о промышленных товарах; выводить общую стоимость товаров, имеющихся на складе.    Поставленную задачу удобно решать с применением технологии наследования при изначальной разработке приложения. По условию задачи, у нас две категории товаров, у которых первые три поля (наименование, номер, цена) совпадают. И действие по добавлению записи тоже совпадает. Таким образом, создадим базовый класс Tovar и два производных от него класса для продуктовых товаров и для промышленных. Производных класс TovarProd для продуктовых товаров будет задействовать все поля и методы базового класса и добавлять еще некоторые специфические именно для себя. Производный класс TovarProm вообще не требует никаких доработок, так как ему полностью хватает возможностей базового класса. Далее создадим массивы для работы с классами продуктовых и промышленных товаров. После создадим обработчик события нажатия на кнопку Button1 и обработчик события нажатия на кнопку Button2, прописав в них логику добавления товаров в соответствующие массивы и вывода записей в Memo1. В обработчик события нажатия кнопки Button3 запишем логику подсчета общей стоимости добавленных товаров. |
| Лабораторная работа № 7.3 | Выполнение лабораторной работы. | Классы. Наследование и отработка взаимодействия с компонентом StringGrid.  Лабораторная работа посвящена изучению классов в C++. Необходимо изучить и повторить проиллюстрированный пример, демонстрирующий применение технологии наследования при изначальном проектировании приложения. Взяв за основу проект из предыдущей лабораторной работы, мы создадим его улучшенную версию, которая позволит добавлять товары не в компонент Memo, а в компонент StringGrid. Также реализуем сохранение данных в текстовый файл и считывание их из него в компонент StringGrid.    Поставленную задачу будем решать с применением технологии наследования при изначальной разработке приложения. Создадим базовый класс и распишем производные классы. Создадим массивы для работы с классами продуктовых и промышленных товаров. После создадим обработчик события нажатия на кнопку Button1 и обработчик события нажатия на кнопку Button2, прописав в них логику добавления товаров в соответствующие массивы и вывода записей в StirngGrid1. В обработчик события нажатия кнопки Button3 запишем логику подсчета общей стоимости добавленных товаров. Далее, создадим обработчик события OnCreate для формы и обработчики нажатия кнопок для сохранения данных в текстовый файл и их загрузки из файла. В последней процедуре Button5Click, создается объект SL, у которого тип данных - StringList. Это строковый список. Удобная конструкция для построчного считывания из файла и заполнения строк таблицы StrinGrid этими данными. Реализовав рассмотренный пример, необходимо добавить возможность правильной работы кнопки "Общая стоимость", если данные не добавлялись вручную, а были загружены из файла. Сейчас же, если запустить программу и загрузить данные из файла кнопкой "Загрузить", то программа выдаст сообщение "Нет товаров!". Это верно, так как при заполнении табличной части компонента StringGrid, мы не записываем данные в массивы товаров, соответственно массивы остаются пустыми. Это надо исправить. Дополните процедуру Button5Click и создайте дополнительный метод класса, позволяющий добавлять данные из StringGrid. То есть, можно сделать по аналогии с методом, который добавляет записи из эдитов:  void dobav\_zap() { // метод для добавления записей  if(VidTovara==0) {  strcpy(name, AnsiString(Form1->LabeledEdit1->Text).c\_str());  number = StrToInt(Form1->LabeledEdit2->Text);  price = StrToFloat(Form1->LabeledEdit3->Text);  }  if(VidTovara==1) {  strcpy(name, AnsiString(Form1->LabeledEdit6->Text).c\_str());  number = StrToInt(Form1->LabeledEdit7->Text);  price = StrToFloat(Form1->LabeledEdit8->Text);  }  }    Необходимо разработать метод, в котором они добавлялись бы из StringGrid. |
| Лабораторная работа № 8.1 | Выполнение лабораторной работы. | Классы. Наследование и виртуальные методы.  Лабораторная работа посвящена изучению классов в ООП на C++. Необходимо изучить и повторить пример, демонстрирующий применение технологии наследования с виртуальными методами. Взяв за основу проект из предыдущей лабораторной работы, мы создадим его улучшенную версию, которая позволит работать не с двумя массивами, а с одним универсальным, который может хранить данные как продуктовых товаров, так и промышленных. Также реализуем сохранение данных в текстовый файл и считывание их из него в компонент StringGrid (по аналогии с предыдущей л/р). Также реализуем нормальное функционирование кнопки "Общая стоимость", которая сможет рассчитывать общую стоимость как добавленных вручную товаров, так и загруженных из файла.  Разместим на форме необходимые для UI компоненты как на скриншоте ниже.    В .h-файле пропишем классы (базовый и производные). Методы базового класса объявим виртуальными. Выдержка из лекции: "Виртуальные функции позволяют решать прямо в процессе выполнения программы, какую именно функцию вызывать. Виртуальные функции дают большую гибкость при выполнении одинаковых действий над разнородными объектами. В частности, они разрешают использование функций, вызванных из массива указателей на базовый класс, который на самом деле содержит указатели (или ссылки) на множество порождённых классов. Это пример полиморфизма."  Переключимся на unit.cpp и распишем все методы и глобальные переменные, как в программном коде ниже.  int VidTovara = 0; // 0-продуктовые, 1-промышленные  int i=0, ij=0; // глобальные переменные счетчики  void Tovar::dobav\_zap() { // метод для добавления записей  if(VidTovara==0) {  strcpy(name, AnsiString(Form1->LabeledEdit1->Text).c\_str());  number = StrToInt(Form1->LabeledEdit2->Text);  price = StrToFloat(Form1->LabeledEdit3->Text);  }  if(VidTovara==1) {  strcpy(name, AnsiString(Form1->LabeledEdit6->Text).c\_str());  number = StrToInt(Form1->LabeledEdit7->Text);  price = StrToFloat(Form1->LabeledEdit8->Text);  }  }  void Tovar::dobav\_zap\_iz\_SG() {  strcpy(name, AnsiString(Form1->StringGrid1->Cells[0][ij]).c\_str());  number = StrToInt(Form1->StringGrid1->Cells[1][ij]);  price = StrToFloat(Form1->StringGrid1->Cells[2][ij]);  }  void Tovar::show() { // метод для показа записей в стринггриде  Form1->StringGrid1->Cells[0][i] = AnsiString(name);  Form1->StringGrid1->Cells[1][i] = number;  Form1->StringGrid1->Cells[2][i] = price;  }  float Tovar::get\_price() { // чтобы не использовать прямой доступ к полю price  return price;  }  void Tovar::clear() { // метод очистки эдитов  if(VidTovara == 0) {  Form1->LabeledEdit1->Clear();  Form1->LabeledEdit2->Clear();  Form1->LabeledEdit3->Clear();  }  if(VidTovara == 1) {  Form1->LabeledEdit6->Clear();  Form1->LabeledEdit7->Clear();  Form1->LabeledEdit8->Clear();  }  }  void TovarProd::dobav\_zap() {  Tovar::dobav\_zap(); // вызов метода базового класса для первых трех полей  srok = StrToInt(Form1->LabeledEdit4->Text);  temp = StrToInt(Form1->LabeledEdit5->Text);  }  void TovarProd::dobav\_zap\_iz\_SG() {  Tovar::dobav\_zap\_iz\_SG(); // вызов метода базового класса  srok = StrToInt(Form1->StringGrid1->Cells[3][ij]);  temp = StrToInt(Form1->StringGrid1->Cells[4][ij]);  }  void TovarProd::show() {  Tovar::show(); // вызов метода базового класса для первых трех полей  Form1->StringGrid1->Cells[3][i] = srok;  Form1->StringGrid1->Cells[4][i] = temp;  }  void TovarProd::clear() {  Tovar::clear();  Form1->LabeledEdit4->Clear();  Form1->LabeledEdit5->Clear();  }  const int K = 100; // максимальное число товаров  Tovar \*prod\_and\_prom[K]; // !!! массив может хранить продтовары и промтовары !!! - одно из главных достоинств виртуальных методов  Tovar \*new\_prom\_and\_prod[K]; // новый массив пром и прод товаров (для стрингрида, заполненного из файла)  Распишите обработчики добавления товаров. Далее, загружаем данные из файла в список StringList, из которого заполняем StringGrid. Разбираем считываемые данные на принадлежность к промышленным или продуктовым товарам и заполняем новый массив, который создали предварительно именно для этих целей.  Реализовав рассмотренный пример, необходимо добавить возможности: а) сортировки товаров по наименованию и цене; б) поиска (по всем свойствам товаров). |

## Критерии, шкалы оценивания текущего контроля успеваемости:

| **Наименование оценочного средства (контрольно-оценочного мероприятия)** | **Критерии оценивания** | **Шкалы оценивания** | |
| --- | --- | --- | --- |
| **100-балльная система** | **Пятибалльная система** |
| Лабораторная работа | Работа выполнена полностью. Нет ошибок в логических рассуждениях и в реализации задания в виде файла или выполняемой программы. Возможно наличие одной неточности или описки, не являющиеся следствием незнания или непонимания учебного материала и не влияющей на функциональные качества программы. Обучающийся показал полный объем знаний, умений в освоении пройденных тем и применение их на практике. Работа зачтена. |  | 5 |
| Работа выполнена полностью, но выбран неэффективный алгоритм или метод реализации, обоснований шагов решения недостаточно. Допущена одна ошибка или два-три недочета, которые незначительно влияют на качество представленной работы. Работа зачтена. |  | 4 |
| Допущены более одной ошибки или более двух-трех недочетов, которые оказывают значительное влияние на представляемый файл или компьютерную программу, ухудшают их информативность и функциональные возможности. Работа зачтена. |  | 3 |
| Работа выполнена не полностью. Допущены грубые ошибки. Файлы не содержат необходимой информации, компьютерная программа выдаёт неправильные результаты при вычислении тестовых примеров. Работа не зачтена. |  | 2 |
| Работа не выполнена. |  |

## Промежуточная аттестация:

|  |  |
| --- | --- |
| **Форма промежуточной аттестации** | **Типовые контрольные задания и иные материалы**  **для проведения промежуточной аттестации:** |
| Зачет с оценкой:  Компьютерное тестирование | Вопрос 1.  Основная часть языка программирования С является подмножеством С++, а большинство программ, написанных на С, являются также программами на С++.   * Обратное утверждение верно * Обратное утверждение неверно   Вопрос 14.  Одной из часто используемых библиотечных функций для работы со строковыми массивами типа char является функция strncmp(s1, s2, n), которая   * копирует не более n символов из строки s2 в s1 и возвращает s1 * добавляет не более n символов из строки s2 к s1 и возвращает s1 * сравнивает строку s1 и первые n символов строки s2   Вопрос 15.  В определении, в объявлении и при вызове одной и той же функции типы и порядок следования параметров   * должны совпадать * могут частично не совпадать * должны различаться   Вопрос 17.  Способ передачи аргументов, при котором функция создаёт копии передаваемых значений, называется   * передачей аргументов по ссылке * передачей аргументов по указателю * передачей аргументов по значению   Вопрос 19.  Допустим, в программе используется некоторая функция вида void function\_1(float& v) { v = v\* 2.54; }. Какой вызов этой функции должен быть в главной функции main() для расчета значения вещественной переменной var?   * function\_1(&var); * function\_1(var);   Вопрос 80.  Политика основных принципов ООП такова, что, если функция является членом класса, она [[1]] доступ к полям класса.  Вопрос 101.  Если поле данных класса описано с ключевым словом static, то значение этого поля будет [[1]] для всех объектов данного класса.  ... |

## Критерии, шкалы оценивания промежуточной аттестации учебной дисциплины:

| **Форма промежуточной аттестации** | **Критерии оценивания** | **Шкалы оценивания** | | |
| --- | --- | --- | --- | --- |
| **Наименование оценочного средства** | **100-балльная система** | **Пятибалльная система** | |
| Зачет с оценкой:  компьютерное тестирование | За выполнение каждого тестового задания испытуемому выставляются баллы. За полностью правильный ответ к каждому заданию с выбором одного правильного варианта выставляется один балл, за неправильный — ноль. За задания с выбором нескольких правильных ответов или в заданиях с сопоставлениями испытуемый может получить менее 1 балла. Например, если правильных ответов в задании два, то за каждый он получает 0,5 балла, если правильных ответов три, то за каждый он получает 0,333 балла и т.п.  Правила оценки всего теста:  вне зависимости от количества заданий в тесте общая сумма баллов за все правильные ответы пересчитывается тестирующей компьютерной системой в итоговые баллы. 10 итоговых баллов эквивалентны 100% правильных ответов. Для того, чтобы получить отличную, хорошую, удовлетворительную или неудовлетворительную оценки, итоговые баллы за промежуточные аттестации каждого семестра складываются с баллами за выполненные лабораторные работы. |  | *5* | *85% - 100%* |
|  | *4* | *65% - 84%* |
|  | *3* | *41% - 64%* |
|  | *2* | *40% и менее 40%* |

## Система оценивания результатов текущего контроля и промежуточной аттестации.

Оценка по дисциплине выставляется обучающемуся с учётом результатов текущей и промежуточной аттестации.

|  |  |  |
| --- | --- | --- |
| **Форма контроля** | **100-балльная система** | **Пятибалльная система** |
| Текущий контроль: |  |  |
| Выполнение лабораторной работы |  | зачтено/не зачтено |
| Промежуточная аттестация  зачёт с оценкой |  | отлично  хорошо  удовлетворительно  неудовлетворительно |
| **Итого за семестр** (дисциплину)  зачёт с оценкой |  |

* + - 1. Полученный совокупный результат конвертируется в пятибалльную систему оценок в соответствии с таблицей:

|  |  |  |
| --- | --- | --- |
| **100-балльная система** | **пятибалльная система** | |
| **зачет с оценкой/экзамен** | **зачет** |
|  | отлично  зачтено (отлично) |  |
|  | хорошо  зачтено (хорошо) |
|  | удовлетворительно  зачтено (удовлетворительно) |
|  | неудовлетворительно |  |

# ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

* + - 1. Реализация программы предусматривает использование в процессе обучения следующих образовательных технологий:
    - проблемная лекция;
    - проектная деятельность;
    - групповые дискуссии;
    - анализ ситуаций и имитационных моделей;
    - преподавание дисциплины на основе результатов научных исследований;
    - поиск и обработка информации с использованием сети Интернет;
    - дистанционные образовательные технологии;
    - использование на лекционных занятиях видеоматериалов и наглядных пособий;
    - самостоятельная работа в системе компьютерного тестирования.

# ПРАКТИЧЕСКАЯ ПОДГОТОВКА

* + - 1. Практическая подготовка в рамках учебной дисциплины реализуется при проведении лабораторных работ, предусматривающих участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью.

# ОРГАНИЗАЦИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ДЛЯ ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

* + - 1. При обучении лиц с ограниченными возможностями здоровья и инвалидовиспользуются подходы, способствующие созданию безбарьерной образовательной среды: технологии дифференциации и индивидуального обучения, применение соответствующих методик по работе с инвалидами, использование средств дистанционного общения, проведение дополнительных индивидуальных консультаций по изучаемым теоретическим вопросам и практическим занятиям, оказание помощи при подготовке к промежуточной аттестации.
      2. При необходимости рабочая программа дисциплины может быть адаптирована для обеспечения образовательного процесса лицам с ограниченными возможностями здоровья, в том числе для дистанционного обучения.
      3. Учебные и контрольно-измерительные материалы представляются в формах, доступных для изучения студентами с особыми образовательными потребностями с учетом нозологических групп инвалидов:
      4. Для подготовки к ответу на практическом занятии, студентам с ограниченными возможностями здоровья среднее время увеличивается по сравнению со средним временем подготовки обычного студента.
      5. Для студентов с инвалидностью или с ограниченными возможностями здоровья форма проведения текущей и промежуточной аттестации устанавливается с учетом индивидуальных психофизических особенностей (устно, письменно на бумаге, письменно на компьютере, в форме тестирования и т.п.).
      6. Промежуточная аттестация по дисциплине может проводиться в несколько этапов в форме рубежного контроля по завершению изучения отдельных тем дисциплины. При необходимости студенту предоставляется дополнительное время для подготовки ответа на зачете или экзамене.
      7. Для осуществления процедур текущего контроля успеваемости и промежуточной аттестации обучающихся создаются, при необходимости, фонды оценочных средств, адаптированные для лиц с ограниченными возможностями здоровья и позволяющие оценить достижение ими запланированных в основной образовательной программе результатов обучения и уровень сформированности всех компетенций, заявленных в образовательной программе.

# МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

* + - 1. Характеристика материально-технического обеспечения дисциплины соответствует требованиям ФГОС ВО.
      2. Материально-техническое обеспечение дисциплины при обучении с использованием традиционных технологий обучения.

| **Наименование учебных аудиторий, лабораторий, мастерских, библиотек, спортзалов, помещений для хранения и профилактического обслуживания учебного оборудования и т.п.** | **Оснащенность учебных аудиторий, лабораторий, мастерских, библиотек, спортивных залов, помещений для хранения и профилактического обслуживания учебного оборудования и т.п.** |
| --- | --- |
| **119071, г. Москва, Малый Калужский переулок, дом 1, строение 3** | |
| аудитории для проведения занятий лекционного типа | комплект учебной мебели,  технические средства обучения, служащие для представления учебной информации большой аудитории:   * компьютерная техника (ноутбук/компьютер); * проектор; * экран. |
| аудитории для проведения практических занятий, выполнения лабораторных работ, занятий по практической подготовке, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации | комплект учебной мебели,  технические средства обучения, служащие для представления учебной информации большой аудитории:   * компьютерная техника (ноутбук/компьютер); * проектор; * экран; * персональные компьютеры, подключенные к сети Интернет. |
| **Помещения для самостоятельной работы обучающихся** | **Оснащенность помещений для самостоятельной работы обучающихся** |
| читальный зал библиотеки: | * компьютерная техника, подключение к сети «Интернет» |

* + - 1. Материально-техническое обеспечение учебной дисциплины при обучении с использованием электронного обучения и дистанционных образовательных технологий.

|  |  |  |
| --- | --- | --- |
| **Необходимое оборудование** | **Параметры** | **Технические требования** |
| Персональный компьютер/ ноутбук/планшет,  камера,  микрофон,  динамики,  доступ в сеть Интернет | Веб-браузер | Версия программного обеспечения не ниже: Chrome 72, Opera 59, Firefox 66, Edge 79, Яндекс.Браузер 19.3 |
| Операционная система | Версия программного обеспечения не ниже: Windows 7, macOS 10.12 «Sierra», Linux |
| Веб-камера | 640х480, 15 кадров/с |
| Микрофон | любой |
| Динамики (колонки или наушники) | любые |
| Сеть (интернет) | Постоянная скорость не менее 192 кБит/с |

Технологическое обеспечение реализации программы осуществляется с использованием элементов электронной информационно-образовательной среды университета.

# УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Автор(ы)** | **Наименование издания** | **Вид издания (учебник, УП, МП и др.)** | **Издательство** | **Год**  **издания** | **Адрес сайта ЭБС**  **или электронного ресурса *(заполняется для изданий в электронном виде)*** | **Количество экземпляров в библиотеке Университета** |
| 10.1 Основная литература, в том числе электронные издания | | | | | | | |
| 1 | Синаторов С.В. | Информационные технологии | Учебное пособие | М.: Флинта | 2021 | <https://znanium.com/catalog/document?id=374932> | - |
| 2 | Шитов В.Н. | Информатика и информационно-коммуникационные технологии в профессиональной деятельности | Учебное пособие | М: НИЦ ИНФРА-М | 2022 | <https://znanium.com/catalog/document?id=388696> | - |
| 3 | Шуляк О.А. | Основы программирования | Учебно-методическая литература | М.: Флинта | 2021 | <https://znanium.com/catalog/document?id=390158> | - |
| 4 | Немцова Т.И., Голова С.Ю., Терентьев А.И.; под ред. Л.Г. Гагариной. | Программирование на языке высокого уровня. Программирование на языке C++ | Учебное пособие | М.: ИД ФОРУМ: ИНФРА-М | 2021 | <https://znanium.com/catalog/document?id=363426> | - |
| 10.2 Дополнительная литература, в том числе электронные издания | | | | | | | |
| 1 | Гутгарц Р. Д. | Проектирование автоматизированных систем обработки информации и управления | Учебное пособие | М.: Издательство Юрайт | 2022 | <https://urait.ru/bcode/494408> | - |
| 2 | Плотникова Н.Г. | Информатика и информационно-коммуникационные технологии (ИКТ) | Учебное пособие | М.: РИОР | 2021 | <https://znanium.com/catalog/document?id=370445> | - |
| 3 | Горбатов С.М., Тарасов Ю.С., Наумова М.Г. | Информационные технологии | Учебное пособие | М.: МИСиС | 2016 | <https://znanium.com/catalog/document?id=371025> | - |
| 4 | Федотова Е.Л. | Информационные технологии и системы | Учебное пособие | М.: Издательский Дом ФОРУМ | 2022 | <https://znanium.com/catalog/document?id=386738> | - |
| 5 | М. В. Огнева, Е. В. Кудрина | Программирование на языке С++: практический курс | Учебное пособие | М.: Издательство Юрайт | 2022 | <https://urait.ru/bcode/492984> | - |
| 10.3 Методические материалы (указания, рекомендации по освоению дисциплины (модуля) авторов РГУ им. А. Н. Косыгина) | | | | | | | |
| 1 | Семенов А.А. | Основы объектно-ориентированного программирования в среде С++Builder | Методическое пособие | М.: ИИЦ МГУДТ | 2010 | локальная сеть университета | 5 |

# ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ УЧЕБНОГО ПРОЦЕССА

## Ресурсы электронной библиотеки, информационно-справочные системы и профессиональные базы данных:

|  |  |
| --- | --- |
| **№ пп** | **Электронные учебные издания, электронные образовательные ресурсы** |
|  | ЭБС «Лань» <http://www.e.lanbook.com/> |
|  | «Znanium.com» научно-издательского центра «Инфра-М»  <http://znanium.com/> |
|  | Электронные издания «РГУ им. А.Н. Косыгина» на платформе ЭБС «Znanium.com» <http://znanium.com/> |
|  | ЭБС «ИВИС» <http://dlib.eastview.com/> |
|  | Образовательная платформа «ЮРАЙТ»  <https://urait.ru/> |
|  | **Профессиональные базы данных, информационные справочные системы** |
|  | Scopus https://www.scopus.com (международная универсальная реферативная база данных, индексирующая более 21 тыс. наименований научно-технических, гуманитарных и медицинских журналов, материалов конференций примерно 5000 международных издательств); |
|  | Научная электронная библиотека еLIBRARY.RU https://elibrary.ru (крупнейший российский информационный портал в области науки, технологии, медицины и образования); |
|  | База данных в мире Academic Search Complete - обширная полнотекстовая научно-исследовательская. Содержит полные тексты тысяч рецензируемых научных журналов по химии, машиностроению, физике, биологии. <http://search.ebscohost.com> |

## Перечень программного обеспечения

|  |  |  |
| --- | --- | --- |
| **№п/п** | **Программное обеспечение** | **Реквизиты подтверждающего документа/ Свободно распространяемое** |
|  | Windows 10 Pro, MS Office 2019 | контракт № 18-ЭА-44-19 от 20.05.2019 |
|  | Microsoft Visual Studio | контракт № 18-ЭА-44-19 от 20.05.2019 |
|  | Embarcadero C++Builder RAD Studio Professional Academic Concurrent License | № 15-02.01-2459 от 21.12.2021  Embarcadero License Certificate: #546431, #546432, #546433, #546434, #546435 |
|  | Code::Blocks — свободная кроссплатформенная среда разработки на C++. | Свободно распространяемое на условиях GNU General Public License v.3. |

### ЛИСТ УЧЕТА ОБНОВЛЕНИЙ РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

В рабочую программу учебной дисциплины внесены изменения/обновления и утверждены на заседании кафедры:

|  |  |  |  |
| --- | --- | --- | --- |
| **№ пп** | **год обновления РПД** | **характер изменений/обновлений**  **с указанием раздела** | **номер протокола и дата заседания**  **кафедры** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |