Автономная некоммерческая организация высшего образования
“ИНСТИТУТ БИЗНЕСА И ДИЗАЙНА”
ФАКУЛЬТЕТ УПРАВЛЕНИЯ БИЗНЕСОМ

УТВЕРЖДАЮ
Ректор _____ С.С. Юров
«24» февраля 2022 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.24 ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Для направления подготовки:
09.03.02 Информационные системы и технологии
(уровень бакалавриата)

Типы задач профессиональной деятельности:
производственно-технологический; организационно-управленческий; проектный.

Направленность (профиль):
Информационные системы и технологии в бизнесе

Форма обучения:
очная, заочная

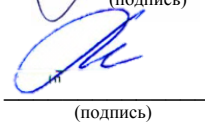
Москва – 2022

Разработчик: Мелехов Игорь Сергеевич, преподаватель кафедры гуманитарных и естественно-научных дисциплин АНО ВО «Институт бизнеса и дизайна».

«15» января 2022 г.  /И.С.Мелехов/

СОГЛАСОВАНО:

Декан факультета  /Н.Е. Козырева /
(подпись)

Заведующий кафедрой
разработчика РПД  /Е.С.Мальцева /
(подпись)

Протокол заседания кафедры № 6 от «27» января 2022 г.

1. ЦЕЛЬ И ЗАДАЧИ ДИСЦИПЛИНЫ

Цель:
изучение теоретических основ современных технологий программирования и получение практических навыков их реализации.

Задачи:
формирование систематизированного представления о концепциях, моделях и принципах организации, положенных в основу "классических" технологий программирования и современных семейств технологий, получение практической подготовки в области выбора и применения технологии программирования для задач автоматизации обработки информации, выработка оценки современного состояния иперспективных направлений развития технологий программирования.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

2.1. Место дисциплины в учебном плане:

Блок: Блок 1. Дисциплины (модули).
Часть: Обязательная часть.
Осваивается: 3-6 семестр очная форма обучения, 4-7 заочная форма обучения.

3. КОМПЕТЕНЦИИ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ

ОПК- 6 – способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий.
ПК - 1 – способность выполнять интеграцию программных модулей и компонент
ПК - 3 – способность выполнять работы по созданию (модификации) и сопровождениюинформационных систем.
ПК - 7 – способность проводить анализ требований к программному обеспечению, выполнять работы по проектированию программного обеспечения.

4. РЕЗУЛЬТАТЫ ОСВОЕНИЯ ДИСЦИПЛИНЫ ОБУЧАЮЩИМСЯ

Код и наименование компетенции	Индикаторы достижения компетенции	Результаты обучения
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	**ОПК-6.1** Знать: методы алгоритмизации, языки и технологии программирования, пригодные дляпрактического применения в области информационных систем итехнологий **ОПК-6.2** Уметь: применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач вобласти информационных систем и технологий **ОПК-6.3** Иметь навыки: программирования, отладки и тестирования прототипов программно-технических комплексов задач	**Знает:** методы алгоритмизации, языки и технологии программирования, пригодные дляпрактического применения в области информационных систем итехнологий **Умеет:** применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач в области информационных систем и технологий **Владеет:** навыками: программирования, отладки и тестирования прототипов программно-технических комплексов задач
ПК-1 Способность выполнять интеграцию программных модулей и компонент	**ПК-1.1** Знать: методы и способы интеграции программных модулейи компонент **ПК-1.2** Уметь: выполнять интеграцию программных модулей и компонент **ПК-1.3** Владеть: методами и способами выполнения интеграции программных	**Знает:** методы и способы интеграции программных модулейи компонент **Умеет:** выполнять интеграцию программных модулей и компонент **Владеет:**

	модулей и компонент	методами и способами выполнения интеграции программных модулей и компонент
ПК-3 Способность выполнять работы по созданию (модификации) и сопровождению информационных систем	**ПК-3.1** Знать: методы и способы по созданию (модификации) и сопровождению информационныхсистем **ПК-3.2** Уметь: выполнять работы по созданию (модификации) и сопровождению информационных систем **ПК-3.3** Владеть: методами и способами выполнения работ по созданию (модификации) и сопровождению информационных систем	**Знает:** методы и способы по созданию (модификации) и сопровождению информационныхсистем **Умеет:** выполнять работы по созданию (модификации) и сопровождению информационныхсистем **Владеет:** методами и способами выполнения работ по созданию (модификации) и сопровождениюинформационных систем
ПК-7 Способность проводить анализ требований к программному обеспечению, выполнять работы по проектированию программного обеспечения	**ПК-7.1** Знать: требования к программному обеспечению, технологии по проектированию программного обеспечения **ПК-7.2** Уметь: проводить анализ требований к программному обеспечению, выполнять работы попроектированию программного обеспечения **ПК-7.3** Владеть: навыками проведенияанализа требований к программному обеспечению, выполнению работы по проектированию программного обеспечения	**Знает:** требования к программномуобеспечению, технологии по проектированию программного обеспечения **Умеет:** проводить анализ требований к программному обеспечению, выполнять работы попроектированию программного обеспечения **Владеет:** навыками проведенияанализа требований к программному обеспечению, выполнению работы по проектированию программного обеспечения

5. ОБЪЕМ ДИСЦИПЛИНЫ И РАСПРЕДЕЛЕНИЕ ВИДОВ УЧЕБНОЙ РАБОТЫ ПО СЕМЕСТРАМ

Общая трудоемкость дисциплины «Технология программирования» для студентов всех форм обучения, реализуемых в АНО ВО «Институт бизнеса и дизайна» по направлению подготовки 09.03.02 Информационные системы и технологии: 22 з.е. / 792 час.

Вид учебной работы	Всего число часов и (или) зачетных единиц (по формам обучения)	
	Очная	Заочная
Аудиторные занятия	340	18
в том числе:		
Лекции	136	32
Практические занятия	204	48
Лабораторные работы	-	
Самостоятельная работа	380	686
в том числе:		
часы на выполнение КР / КП	-	-
Промежуточная аттестация:		
Вид	2 зачета с оценкой 2 экзамена	2 зачета с оценкой 2 экзамена
Трудоемкость (час.)	-	26
Общая трудоемкость з.е. / часов	22 з.е. / 792 час.	22 з.е. / 792 час.

6. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Темы дисциплины		Количество часов (по формам обучения)							
		Очная				Заочная			
№	Наименование	Лекции	Практические занятия	Лабораторные работы	Самост. работа *(в т.ч. КР / КП)*	Лекции	Практические занятия	Лабораторные работы	Самост. работа *(в т.ч. КР / КП)*
1	Программное средство как продукт технологии программирования	12	18		30	2	2		52
2	Основные этапы процесса проектирования программного обеспечения	12	18		30	3	5		52
3	Стандарты и методики, используемые при разработке программных средств	12	18		30	3	5		52
	Итого за семестр (часов)	36	54		90	8	12		156
	Форма контроля:	*Зачет с оценкой*				*Зачет с оценкой*			4
4	Методы проектирования и разработки программного обеспечения	16	24		50	4	6		93
5	Методология объектно-ориентированного программирования	16	24		50	4	6		94
	Итого за семестр (часов)	32	48		100	8	12		187
	Форма контроля:	*Экзамен*			36	*Экзамен*			9
6	Проектирование библиотек классов	12	18		30	2	2		52
7	Проектирование интерфейса с пользователем	12	18		30	3	5		52
8	Технологические средства разработки программного обеспечения	12	18		30	3	5		52
	Итого за семестр (часов)	36	54		90	8	12		156
	Форма контроля:	*Зачет с оценкой*				*Зачет с оценкой*			4
9	Технологии коллективной разработки программного обеспечения	16	24		50	4	6		93
10	Методы отладки и тестирования программ	16	24		50	4	6		94
	Итого за семестр (часов)	32	48		100	8	12		187
	Форма контроля:	*Экзамен*			36	*Экзамен*			9
	Итого (часов) Дисциплина / форма контроля	136	204		380 / 72	32	48		686 / 26
	Всего по дисциплине:	792 / 22 з.е.				792 / 22 з.е.			

СОДЕРЖАНИЕ ТЕМ ДИСЦИПЛИНЫ

Тема 1. Программное средство как продукт технологии программирования.

Основные понятия и термины. Программа. Характеристика программ. Понятие программного обеспечения. Виды программного обеспечения. Программный продукт. Характерные особенности программного продукта. Программный комплекс. Программное средство. Определение требований к программному средству. Программная система. Сложность и сложные системы. Источники сложности. Признаки работоспособной сложной системы. Классификация программных систем по сложности. Проблемы проектирования сложных программных средств.

Тема 2. Основные этапы процесса проектирования программного обеспечения.

Жизненный цикл программного обеспечения. Управление проектом, планирование и распределение ресурсов, контроль исполнения сроков. Тестирование и оценка качества. Управление программными конфигурациями. Сопровождение. Модернизация и масштабирование программного обеспечения.

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

Виды стандартов. Методики проектирования. Стандартизация жизненного цикла программного средства в системе государственных стандартов и стандартов ISO. Стандарт пользовательского интерфейса. Стандарт проектирования. Стандарт оформления документации. Стандартизация процесса разработки программ и программной документации. Перспективы развития технологии программирования, автоматизированного проектирования программных систем на основе языков новых поколений. Доказательное программирование и визуальное программирование.

Тема 4. Методы проектирования и разработки программного обеспечения.

Обзор методов проектирования и разработки программного обеспечения. Основные подходы к разработке программ: процедурное, логическое, функциональное объектно-ориентированное программирование. Процедурное и декларативное представление данных. Связь структур данных и алгоритмов их обработки с управляющими структурами языков программирования. Типы данных в языках программирования. Связь между данными и операциями. Абстрактные типы данных. Понятие интерфейса.

Тема 5. Методология объектно-ориентированного программирования.

Основные положения объектного подхода к разработке программ. Принципы объектного подхода: абстрагирование, ограничение доступа, модульность, иерархия, типизация, параллелизм, устойчивость. Способы распределения и заимствования шаблонов и поведения. Инкапсуляция и спецификация правил доступности элементов класса. Конструкторы и деструкторы. Наследование. Иерархия классов. Одиночное и множественное наследование. Способы реализации множественного наследования, их достоинства и недостатки. Таблицы виртуальных функций. Абстрактные базовые классы. Конечные производные классы. Отношения между объектами и/или классами. Полиморфизм. Ранее и позднее связывание. Управление последовательностью действий в объектно-ориентированной программе. Объект и процесс. Инициализация и взаимодействие объектов и процессов. Сообщения. Реализация механизмов посылки сообщений. Примеры функционирования объектно-ориентированной программы.

Тема 6. Проектирование библиотек классов.

Виды классов: конкретный тип, абстрактный тип, узловой класс, интерфейсный класс. Динамическая идентификация типа. Управление видимостью и областью действия имен. Управление памятью. Библиотеки контейнерных классов. Номенклатура контейнеров и примеры их использования. Иерархия классов исключений.

Тема 7. Проектирование интерфейса с пользователем.

Библиотеки интерфейсных элементов. Понятие приложения. Диалоговые окна и дочерние элементы управления.

Тема 8. Технологические средства разработки программного обеспечения.

Языки программирования четвертого поколения, CASE-системы, системы ускоренной разработки приложений. Системный анализ. Принципы объектно-ориентированного анализа и их обсуждение. Язык объектного моделирования UML. Основные определения: система, домен, подсистема, элемент, связи, среда. Структура системы, декомпозиция, иерархия элементов. Процессы в системе и потоки

информации. Исследование действий. Построение моделей доменов и подсистем, связей и взаимодействия подсистем, взаимодействия объектов, событий, процессов, потоков данных, действий. Описание классов и их взаимосвязей. Динамика поведения объектов, диаграммы перехода состояний. Диаграммы объектов. Видимость и синхронизация объектов, временные диаграммы. Диаграмма процессов. Обработка исключительных ситуаций. Рабочие продукты, методологии и средства анализа и проектирования.

Тема 9. Технологии коллективной разработки программного обеспечения.

Обзор и классификация средств поддержки коллективной разработки программного обеспечения. Программные средства планирования и управления процессом разработки. Сетевые графики и диаграммы рабочего процесса. Сценарии выполнения работ, согласование графиков. Применение систем управления документами.

Тема 10. Методы отладки и тестирования программ.

Инструментальные средства верификации и тестирования программ. Планирование и автоматизированная генерация тестов. Сценарии тестирования. Анализаторы профиля выполнения теста. Репозиторий тестов. Контроль показателей качества.

7. ПРИМЕРНАЯ ТЕМАТИКА КУРСОВЫХ РАБОТ

Курсовая работа не предусмотрена

8. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ: Приложение 1.

9. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ:

9.1. Рекомендуемая литература:

1. Златопольский, Д. М. Программирование: типовые задачи, алгоритмы, методы: учебное пособие: [12+] / Д. М. Златопольский. – 4-е изд. (эл.). – Москва: Лаборатория знаний, 2020. – 226 с.

Режим доступа: https://biblioclub.ru/index.php?page=book_red&id=222873

2. КолесниковаТатьяна, Г. Языки программирования: учебное пособие: [16+] / Г. КолесниковаТатьяна ; Кемеровский государственный университет. – Кемерово: Кемеровский государственный университет, 2019. – 182 с.

Режим доступа: https://biblioclub.ru/index.php?page=book_red&id=573802

3. Мирошниченко, И. И. Языки и методы программирования: учебное пособие: [16+] / И. И. Мирошниченко, Е. Г. Веретенникова, Н. Г. Савельева; Ростовский государственный экономический университет (РИНХ). – Ростов-на-Дону: Издательско-полиграфический комплекс РГЭУ (РИНХ), 2019. – 188 с.

Режим доступа: https://biblioclub.ru/index.php?page=book_red&id=567706

9.2. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень лицензионного и свободно распространяемого программного обеспечения.

При осуществлении образовательного процесса по данной учебной дисциплине предполагается использование:

Лицензионное и свободно распространяемое программное обеспечение, в том числе отечественного производства:

1. Windows 10 Pro Professional (Договор: Tr000391618, срок действия с 20.02.2020 г. по 28.02.2023 г., Лицензия: V8732726);
2. Microsoft Office Professional Plus 2019 (Договор: Tr000391618, срок действия с 20.02.2020 г. по 28.02.2023 г., Лицензия: V8732726).
3. Браузер Google Chrome;
4. Браузер Yandex;
5. Adobe Reader - программа для просмотра, печати и комментирования документов в формате

PDF

9.3. Перечень современных профессиональных баз данных, информационных справочных систем и ресурсов информационно-телекоммуникационной сети «Интернет»

1. https://biblioclub.ru/ - университетская библиотечная система online Библиоклуб.ру
2. http://window.edu.ru/ - единое окно доступа к образовательным ресурса
3. https://uisrussia.msu.ru/ - база данных и аналитических публикаций университетской информационной системы Россия
4. https://www.elibrary.ru/ - электронно-библиотечная система eLIBRARY.RU, крупнейшая в России электронная библиотека научных публикаций
5. http://www.consultant.ru/ - справочная правовая система КонсультантПлюс
6. https://gufo.me/ - справочная база энциклопедий и словарей
7. https://slovaronline.com - поисковая система по всем доступным словарям и энциклопедиям
8. https://www.tandfonline.com/ - коллекция журналов Taylor&Francis Group включает в себя около двух тысяч журналов и более 4,5 млн. статей по различным областям знаний
9. https://openedu.ru - «Национальная платформа открытого образования» (ресурсы открытого доступа)
10. https://www.rsl.ru - Российская Государственная Библиотека (ресурсыоткрытого доступа)
11. https://link.springer.com - Международная реферативная база данных научных изданий Springerlink (ресурсы открытого доступа)
12. https://zbmath.org - Международная реферативная база данных научных изданий zbMATH (ресурсы открытого доступа)

10. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

1. Оборудованные учебные аудитории, в том числе с использованием видеопроектора и подключением к сети «Интернет» и доступом в электронную информационно-образовательную среду Института.
2. Аудитории для самостоятельной работы с подключением к сети «Интернет» и доступом в электронную информационно-образовательную среду Института.
3. Компьютерный класс с подключением к сети «Интернет» и доступом в электронную информационно-образовательную среду Института.
4. Аудио и видеоаппаратура.

№ 403

Учебная аудитория для проведения учебных занятий. Аудитория оснащена оборудованием и техническими средствами обучения:

а) учебной мебелью: столы, стулья, доска маркерная учебная

б) стационарный широкоформатный мультимедиа-проектор Epson EB-X41, экран, колонки.

в) 11 компьютеров, подключенных к сети

«Интернет», с обеспечением доступа в электронную информационно- образовательную среду АНО ВО «Институт бизнеса и дизайна»

№ 402

Помещение для самостоятельной работы. Аудитория оснащена оборудованием и техническими средствами обучения:

а) учебной мебелью: столы, стулья, доска маркерная учебная

б) стационарный широкоформатный мультимедиа-проектор Epson EB-X41, экран, колонки.

в) 11 компьютеров, подключенных к сети «Интернет», с обеспечением доступа в электронную информационно-образовательную среду АНО ВО «Институт бизнеса и дизайна»

11. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

В процессе освоения дисциплины студенту необходимо посетить все виды занятий,

предусмотренные рабочей программой дисциплины и выполнить контрольные задания, предлагаемые преподавателем для успешного освоения дисциплины. Также следует изучить рабочую программу дисциплины, в которой определены цели и задачи дисциплины, компетенции обучающегося, формируемые в результате освоения дисциплины и планируемые результаты обучения. Рассмотреть содержание тем дисциплины; взаимосвязь тем лекций и практических занятий; бюджет времени по видам занятий; оценочные средства для текущей и промежуточной аттестации; критерии итоговой оценки результатов освоения дисциплины. Ознакомиться с методическими материалами, программно-информационным и материально техническим обеспечением дисциплины.

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.

Методические рекомендации для обучающихся с ОВЗ и инвалидов

по освоению дисциплины

В АНО ВО «Институт бизнеса и дизайна» созданы специальные условия для получения высшего образования по образовательным программам обучающимися с ограниченными возможностями здоровья (ОВЗ).

Для перемещения инвалидов и лиц с ограниченными возможностями здоровья в АНО ВО «Институт бизнеса и дизайна» созданы специальные условия для беспрепятственного доступа в учебные помещения и другие помещения, а также их пребывания в указанных помещениях с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся.

При получении образования обучающимся с ограниченными возможностями здоровья при необходимости предоставляются бесплатно специальные учебники и учебные пособия, иная учебная литература. Также имеется возможность предоставления услуг ассистента, оказывающего обучающимся с ограниченными возможностями здоровья необходимую техническую помощь, в том числе услуг сурдопереводчиков и тифлосурдопереводчиков.

Получение доступного и качественного высшего образования лицами с ограниченными возможностями здоровья обеспечено путем создания в институте комплекса необходимых условий обучения для данной категории обучающихся. Информация о специальных условиях, созданных для обучающихся с ограниченными возможностями здоровья, размещена на сайте института (https://obe.ru/sveden/ovz/).

Для обучения инвалидов и лиц с ОВЗ, имеющих нарушения опорно-двигательного аппарата обеспечиваются и совершенствуются материально-технические условия беспрепятственного доступа в учебные помещения, столовую, туалетные, другие помещения, условия их пребывания в указанных помещениях (наличие пандусов, поручней, расширенных дверных проемов и др.).

Для адаптации к восприятию обучающимися инвалидами и лицами с ОВЗ с нарушенным слухом справочного, учебного материала, предусмотренного образовательной программой по выбранным направлениям подготовки, обеспечиваются следующие условия:

для лучшей ориентации в аудитории, применяются сигналы, оповещающие о начале и конце занятия (слово «звонок» пишется на доске);

внимание слабослышащего обучающегося привлекается педагогом жестом (на плечо кладется рука, осуществляется нерезкое похлопывание);

разговаривая с обучающимся, педагог смотрит на него, говорит ясно, короткими предложениями, обеспечивая возможность чтения по губам.

Компенсация затруднений речевого и интеллектуального развития слабослышащих инвалидов и лиц с ОВЗ проводится за счет:

использования схем, диаграмм, рисунков, компьютерных презентаций с гиперссылками, комментирующими отдельные компоненты изображения;

регулярного применения упражнений на графическое выделение существенных признаков предметов и явлений;

обеспечения возможности для обучающегося получить адресную консультацию по электронной почте по мере необходимости.

Для адаптации к восприятию инвалидами и лицами с ОВЗ с нарушениями зрения справочного, учебного, просветительского материала, предусмотренного образовательной программой АНО ВО «Институт бизнеса и дизайна» по выбранной специальности, обеспечиваются следующие условия:

ведется адаптация официального сайта в сети Интернет с учетом особых потребностей инвалидов по зрению, обеспечивается наличие крупношрифтовой справочной информации о расписании учебных занятий;

в начале учебного года обучающиеся несколько раз проводятся по зданию АНО ВО «Институт бизнеса и дизайна» для запоминания месторасположения кабинетов, помещений, которыми они будут пользоваться;

педагог, его собеседники, присутствующие представляются обучающимся, каждый раз называется тот, к кому педагог обращается;

действия, жесты, перемещения педагога коротко и ясно комментируются;

печатная информация предоставляется крупным шрифтом (от 18 пунктов), тотально озвучивается; обеспечивается необходимый уровень освещенности помещений;

предоставляется возможность использовать компьютеры во время занятий и право записи объяснения на диктофон (по желанию обучающегося).

Форма проведения текущей и промежуточной аттестации для обучающихся с ОВЗ

определяется преподавателем в соответствии с учебным планом. При необходимости обучающемуся с ОВЗ с учетом его индивидуальных психофизических особенностей дается возможность пройти промежуточную аттестацию устно, письменно на бумаге, письменно на компьютере, в форме тестирования и т.п., либо предоставляется дополнительное время для подготовки ответа.

Приложение 1

Автономная некоммерческая организация высшего образования
«ИНСТИТУТ БИЗНЕСА И ДИЗАЙНА»

Факультет управления бизнесом

Фонд оценочных средств

Текущего контроля и промежуточной аттестации
по дисциплине (модулю)

Б1.О.24 ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Для направления подготовки:
09.03.02 Информационные системы и технологии
(уровень бакалавриата)

Типы задач профессиональной деятельности:
производственно-технологический; организационно-управленческий; проектный.

Направленность (профиль):
Информационные системы и технологии в бизнесе

Форма обучения:
очная, заочная

Москва – 2022

РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Код и наименование компетенции	Индикаторы достижения компетенции	Результаты обучения
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	**ОПК-6.1** Знать: методы алгоритмизации, языки и технологии программирования, пригодные дляпрактического применения в области информационных систем итехнологий **ОПК-6.2** Уметь: применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач вобласти информационных систем и технологий **ОПК-6.3** Иметь навыки: программирования, отладки и тестирования прототипов программно-технических комплексов задач	**Знает:** методы алгоритмизации, языки и технологии программирования, пригодные дляпрактического применения в области информационных систем итехнологий **Умеет:** применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач в области информационных систем и технологий **Владеет:** навыками: программирования, отладки и тестирования прототипов программно-технических комплексов задач
ПК-1 Способность выполнять интеграцию программных модулей и компонент	**ПК-1.1** Знать: методы и способы интеграции программных модулейи компонент **ПК-1.2** Уметь: выполнять интеграцию программных модулей и компонент **ПК-1.3** Владеть: методами и способами выполнения интеграции программных модулей и компонент	**Знает:** методы и способы интеграции программных модулейи компонент **Умеет:** выполнять интеграцию программных модулей и компонент **Владеет:** методами и способами выполнения интеграции программных модулей и компонент
ПК-3 Способность выполнять работы по созданию (модификации) и сопровождению информационных систем	**ПК-3.1** Знать: методы и способы по созданию (модификации) и сопровождению информационныхсистем **ПК-3.2** Уметь: выполнять работы по созданию (модификации) и сопровождению информационных систем **ПК-3.3** Владеть: методами и способами выполнения работ по созданию (модификации) и сопровождению информационных систем	**Знает:** методы и способы по созданию (модификации) и сопровождению информационныхсистем **Умеет:** выполнять работы по созданию (модификации) и сопровождению информационныхсистем **Владеет:** методами и способами выполнения работ по созданию (модификации) и сопровождениюинформационных систем
ПК-7 Способность проводить анализ требований к программному обеспечению, выполнять работы по проектированию программного обеспечения	**ПК-7.1** Знать: требования к программному обеспечению, технологии по проектированию программного обеспечения **ПК-7.2** Уметь: проводить анализ требований к программному обеспечению, выполнять работы попроектированию программного обеспечения **ПК-7.3** Владеть: навыками проведенияанализа требований к программному обеспечению, выполнению работы по проектированию программного обеспечения	**Знает:** требования к программномуобеспечению, технологии по проектированию программного обеспечения **Умеет:** проводить анализ требований к программному обеспечению, выполнять работы попроектированию программного обеспечения **Владеет:** навыками проведенияанализа требований к программному обеспечению, выполнению работы по проектированию программного обеспечения

Типовые оценочные средства, необходимые для оценки планируемых результатов обучения по дисциплине (модулю):

ТЕКУЩИЙ КОНТРОЛЬ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ (МОДУЛЮ)

Тест для формирования «ОПК-6.1»

Вопрос №1 .

Ключевое слово template предназначено в языке С++ 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 А перегружается операция = =, какое должно быть возвращаемое значение функции, осуществляющей перегрузку

Варианты ответов:

1. A
2. bool
3. int
4. void

Критерии оценки выполнения 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

Практическое задание для формирования «ОПК-6.2»

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.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ОПК-6.2»

РАЗРАБОТКА КЛАССА ВЕКТОР

1. Цель работы

Разработать класс вектор для хранения целых чисел с проверкой правильности обращения по индексу и встроенными операциями сложения и вычитания векторов, умножения на скаляр, скалярного произведения.

2. Пример конструирования класса

Для хранения данных потребуется некоторая область памяти, начало которой мы определим как указатель на целое (int* ia;), а размер ее будем хранить в целом числе (int size;). Тогда определение класса может выглядеть следующим образом:

```
const int IntArraySize = 24;
```

```
class IntArray{
public:
IntArray(int sz = IntArraySize);
IntArray(const IntArray&);
~IntArray(){ delete [] ia;};
IntArray& operator=(const IntArray&);
int& operator[](int);
int getSize() const { return size;};
protected:
int size;
int* ia;
};
```

Определенная в строке “const int IntArraySize = 24;” константа используется при конструировании объекта типа IntArraySize с размером по умолчанию: IntArraySize ia; //здесь объект ia будет создан с size = 24

IntArraySize ia1(40); //здесь объект ia1 будет создан с size = 40 .

Конструктор класса может иметь следующий вид:

```
IntArray ::IntArray(int sz) {
size = sz;
ia = new int[size];
}
```

Конструктор копирования класса X::X(const X&) предназначен для конструирования объекта следующим образом:

IntArraySize ia2 = ia1;

Объект ia2 будет создан в виде точной копии объекта ia1, при отсутствии же конструктора копирования будет выполнена почленная инициализация ia2 данными ia1, то есть ia2.size = ia1.size; ia2.ia =ia1.ia; последнее представляет потенциальную опасность. Время существования объектов ai1 и ia2 может быть разным, если ia1 прекращает существование деструктор освобождает память на которую продолжает указывать указатель ia2.ia, поскольку он был инициализирован указателем ia1.ia при почленном копировании.

Таким образом конструктор копирования должен создать новый объект являющийся точной копией объекта которым осуществляется инициализация:

```
IntArray ::IntArray(const IntArray& IA) {
size = IA.size;
ia = new int[size];
for(int i = 0; i < size; I++)
{
ia[i] = IA.ia[i];
}
}
```

Поскольку размер копируемого объекта не обязан совпадать с прежним размером, необходимо освободить память, выделить ее заново для копируемого объекта и перенести данные.

Перегрузка операции “равно” будет использована в случае:

ia = ia1;

Для того чтобы реализовать такую возможность необходимо определить перегрузку операции “равно”:

```
IntArray& IntArray::operator =(const IntArray& iA){
delete [] ia;
size = iA.size;
ia = new int[size];
for(int i = 0; i < size; i++)
{
ia[i] = iA.ia[i];
}
return *this;
}
```

Определение операции []может выглядеть следующим образом:

```
int& IntArray::operator[](int index){
return ia[index];
}
```

В производном от IntArray классе, вероятно потребуется переопределение этой операции таким образом чтобы исключить обращение по неверному индексу.

Для создания нового объекта типа IntArray с одновременной инициализацией с помощью предопределенного массива целых, например таким образом:

```
int arr[] = {1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0};
IntArray arr1(arr,19);
```

потребуется определение еще одного конструктора IntArray(int*,int);.

```
IntArray::IntArray( int *arr, int sz) {
size = sz;
ia = new int[size];
for(int i = 0; i < size; i++) {
ia[i] = arr[i];
}
}
```

Используя построенный класс создадим производный от него класс снабдив его дополнительными свойствами.

```
class IntArrayRC : public IntArray {
public:
IntArrayRC(int = ArraySize);
int& operator [](int);
protected:
void RangeCheck(int);
};
```

Переопределим операцию [] для обеспечения проверки выхода индекса за границы массива.
Реализация операции может выглядеть следующим образом:

```
int& IntArrayRC :: operator[](int index){
RangeCheck(index);
return ia[index];
}
```

Функция член класса RangeCheck(int), для проверки правильности доступа по индексу, может быть

реализована следующим образом:

```
#include <stdlib.h>
void IntArrayRC::RangeCheck(int index) {
if(index <= 0 || index >=size){
cerr <<"Неверный индекс!\n";
exit(1);
}
}
```

3. Задание

Определить класс для хранения целых чисел с проверкой правильности обращения по индексу и встроенными операциями сложения и вычитания векторов, умножения на скаляр, скалярного произведения. Написать функцию main() и проиллюстрировать в ней все определенные операции с объектами класса.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ОПК-6.3»

РАЗРАБОТКА ШАБЛОННОГО КЛАССА ВЕКТОР

1. Цель работы

На базе разработанного в работе N 2 класса создать шаблонный класс с аналогичными свойствами предназначенный для хранения и операций с типами int, short, float, double.

2. Теоретические сведения

C++ строго типизированный язык. В некоторых случаях это кажется помехой для реализации некоторых функций или классов.

Например, алгоритм функции нахождения минимума очевиден, но строгая типизация языка требует, чтобы для каждой пары типов, которые приходится сравнивать, был определен свой вариант функции:

```
int min(int a, int b) {
return a < b ? a : b;
}
double min(double a, double b) {
return a < b ? a : b;
}
```

и так для каждой пары типов.

Возможен конечно вариант использования директив препроцессора:

#define min(a,b) ((a) < (b) ? (a) : (b))

В простых случаях, это хорошо работает, но легко получить неожиданный результат:

while(min(p++,&ia[size]) != &ia[size]) count++;

Механизм шаблонов дает такую же компактную форму записи как и в макроопределениях, но сохраняет строгий контроль типов. Параметризированы могут быть один, несколько или все типы, тело функции остается инвариантным:

template <class T> T min(T a, T b) {

return a < b ? a : b;

}

Автоматическое порождение экземпляров класса, рассчитанных на конкретные типы в C++ также решается с помощью шаблонов:

template<class T> class Array{

Array(int sz = 12) : ia(new T[sz]) , size(sz) {}

~Array() { delete [] ia;}

private :

T* ia;

int size;

};

Теперь для получения объектов массивов целых, комплексных, строк достаточно написать:

Array<int> iA;

Array<complex> iC;

Array<string> iS;

3. Задание

На основе результатов предыдущей работы определить шаблонный класс для хранения различных типов чисел с проверкой правильности обращения по индексу и встроенными операциями сложения и вычитания векторов, умножения на скаляр, скалярного произведения. Написать функцию main() и проиллюстрировать в ней все определенные операции с объектами класса.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Тест для формирования «ПК-1.1»

Вопрос №1 .

Переменным целого типа (int) присвоены следующие значения: a=128, b=64, c=256, d=16; в какой из

переменных установлен в единицу (1) шестой разряд:

Варианты ответов:

1. a
2. b
3. c
4. d

Вопрос №2 .

Как найти квадратный корень из числа x

Варианты ответов:

1. Sqrt(x)
2. Summ.Koren(x)
3. Summ.Koren(x)
4. Math.Sqrt(x)

Вопрос №3 .

Что делает оператор «%»

Варианты ответов:

1. Возвращает процент от суммы
2. Возвращает остаток от деления
3. Возвращает тригонометрическую функцию
4. Ни чего из выше перечисленного

Вопрос №4 .

Что сделает программа, выполнив следующий код: Console.WriteLine(«Hello, World!»);

Варианты ответов:

1. Напишет на новой строчке Hello, World!
2. Напишет Hello, World!
3. Удалит все значения с Hello, World!
4. Вырежет слово Hello, World! из всего текста

Вопрос №5 .

Как сделать инкрементацию числа

Варианты ответов:

1. ++
2. —
3. %%
4. !=

Критерии оценки выполнения 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

Практическое задание для формирования «ПК-1.2»

Создать одномерный массив. Заполнить его с клавиатуры. Найти сумму элементов массива. Вывести элементы, кратные трем

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-1.2»

Элементы одномерного массива и их количество вводятся пользователем. Вывести массив на экран. Найти количество целых и сумму положительных элементов. Вывести номера элементов, целая часть которых равна нулю.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-1.2»

Создать компьютерную программу, предлагающую пользователю ввести с клавиатуры 2 знаковых целых числа: a и b и выводящую их сумму – c.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-1.3»

Задан одномерный массив, элементами которого являются символы, вводимые с клавиатуры. Количество элементов задается пользователем. Вывести на экран массив в прямом и обратном порядке в два столбика. Выписать элементы, стоящие на местах, кратных трем в прямом массиве. Указания: Выводить элементы в двух напрвлениях следует в цикле в одном операторе вывода: wrtiteln(a[i]:6,a[…]:6). Вместо многоточия напишите выражение (зависящее от i), которое бы вычисляло номер элемента массива так, чтобы он нумеровал элементы в обратном порядке. Для того, чтобы выписать элементы, стоящие на местах, кратных трем, нужно в новом цикле задать условие: if i mod 3 =0 then…, так как номер элемента – это переменная i

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-1.3»

Вывести на экран список учащихся группы в два столбика в прямом и обратном порядке, организовав одномерный массив, количество элементов которого задает пользователь. Выписать фамилии учащихся, соящих на четных местах в прямом массиве.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-1.3»

Вычислить значения выражений:

а) round(6.9); в) 20 div 6; д) 20 mod 6;

б) round(6.2); г) 2 div 5; ж) 2 mod 5;

и) 3*7 div 2 mod 7/3-trunc(sin(1))

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Тест для формирования «ПК-3.1»

Вопрос №1 .

Что вернет функция Termin после выполнения. Код:

```
int Termin()
{
int a = 1;
int b = 3;
if (a != 5) return a + b;
else return 0
```

Варианты ответов:

1. 5
2. 3
3. 4

Вопрос №2 .

Какие бывают массивы?

Варианты ответов:

1. Разнообразные
2. Сложные и простые
3. Одномерные и многомерные
4. Резиновые и статичные

Вопрос №3 .

Какой оператор возвращает значение из метода?

Варианты ответов:

1. Veni
2. return
3. out
4. end

Вопрос №4 .

Выберите правильное описание переменных: переменная x для сохранения действительного типа, переменная a для сохранения целого числа, символьная переменная c; переменную c инициализировать значением ‘n’, переменную x инициализировать значением 18.21

Тип ответа: Многие из многих
Варианты ответов:

1. Float x, a; char c; c = ‘n’;
2. Float x = 18.21; a; char c = ‘n’;
3. Float x = 18.21; int a; char c = ‘n’;
4. Float x; x = 18.21; int a; char c = ‘n’;

Вопрос №5 .

Укажите, что может быть индексом элемента массива

Варианты ответов:

1. Символ
2. Дробное число
3. Целое положительное число
4. Математическое выражение, результат вычисления которого является целое положительное число

Критерии оценки выполнения 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

Практическое задание для формирования «ПК-3.2»

Варианты индивидуальных заданий
Используя базовый адрес и функцию CopyMemory прочитайте информацию из отображаемого файла. Согласно варианту измените текстовый файл и запишите информацию в этот же файл:

№ варианта	Задание
1	Представить зеркальное отображение содержимого файла
2	Отсортировать по убыванию содержимое числового файла
3	Отсортировать по возрастанию содержимое числового файла
4	Найти минимальное число в файле, содержащем числа
5	Найти максимальное число в файле, содержащем числа
6	Упорядочить буквы по алфавиту
7	Упорядочить буквы в обратном алфавитном порядке
8	Посчитать количество строчных и прописных букв
9	Посчитать количество русских и латинских букв в тексте
10	Посчитать количество цифр в тексте, содержащем буквенно-цифровую информацию
11	Удалить все буквы а из текста. В конец файла записать, сколько букв было удалено.
12	Заменить все буквы на коды ASCII
13	Заменить все строчные русские буквы на прописные

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-3.2»

Удалить столбец двумерного массива целых чисел, в котором находится максимальный элемент этого массива.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-3.3»

Вывести на экран траекторию движения снаряда при следующих исходных данных:

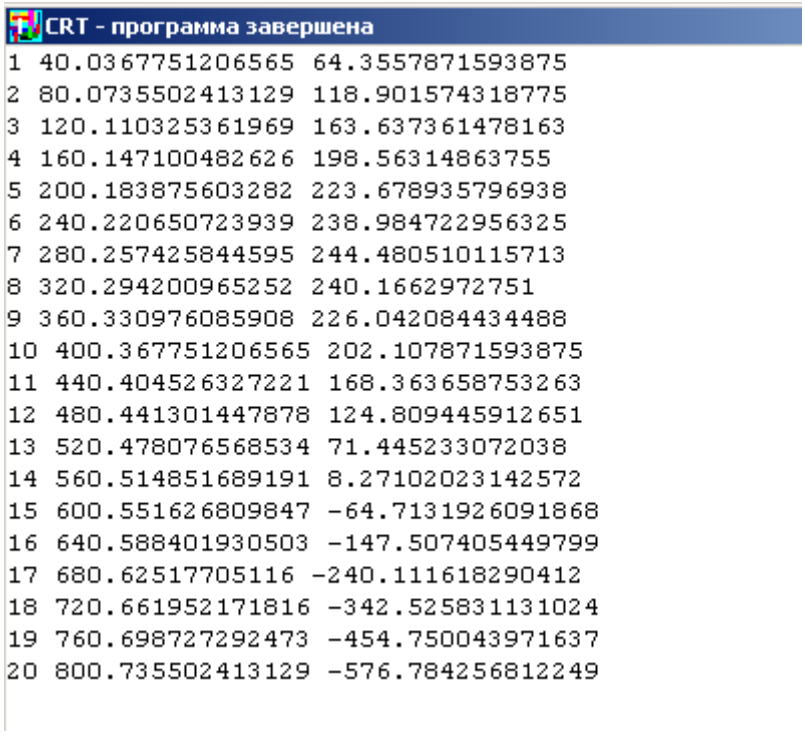
Начальная скорость = 80 м/с

Угол пуска = 60 градусов

Время полета = 20 сек.

Решение:

Программа	Результат выполнения программы

program s1;
uses crt;
var a,v,x,y:real;
i,t:integer;
const g=9.81;
 	
begin
v:=80;
a:=60*3.14/180;
 	
for t:=1 to 20 do
begin
...
writeln(t,' ',x,' ',y);
end;
 	
end.
	
	Рисунок 3

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-3.3»

Создать класс, имитирующий работу программного стека. Конструктор класса выделяет в управляемой куче память под стек, размер которой определяется параметром конструктора. Класс должен содержать функции, имитирующие машинные команды – push, pop. Перегрузить для класса операцию присваивания =. Обосновать необходимость перегрузки этой операции для данного класса.

Осуществить перегрузку операции индексации [] для класса, имитирующего массив целых чисел с контролем границ массива. Конструктор класса выделяет в управляемой куче память, размер которой определяется параметром конструктора

Создать класс, имитирующий работу очереди (дисциплина «first in first out).

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Тест для формирования «ПК-7.1»

Вопрос №1 .

Приведено 4 прототипа (объявления) функций. Выберете вариант, при котором функция изменит значение переданного параметра.

Варианты ответов:

1. int func(int);
2. int & func(int);
3. void func(int&)
4. void func(void)

Вопрос №2 .

struct a

{

int x,y;

};

a v [] = {1,2,3,4,5,6}; В результате инициализации массива v содержимое переменных v [1].x и v [1].y будет равно соответственно

Варианты ответов:

1. 1, 2
2. 2, 3
3. 3, 4
4. 5, 6

Вопрос №3 .

Укажите ошибочный вариант определения функции

Варианты ответов:

1. void func (int a) {cout << a;}
2. int func(int a) {return a +100;}
3. double func (int a) {return a*a*3.14;}
4. void func (int a) { return a*100; }

Вопрос №4 .

```
char* fu1() { return "red"; }
char* fu2() { return "green"; }
char* fu3() { return "blue"; }
int main()
{
char*(*pf[])() = {fu1,fu2,fu3};
    cout << pf[1]();
    cout << pf[0]();
    cout << pf[0];
    cout << pf[1];
}
```

В каком случае на экране отобразится строка “red”

Варианты ответов:

1. cout << pf[1]();
2. cout << pf[0]();
3. cout << pf[0];
4. cout << pf[1];

Вопрос №5 .

Укажите атрибут, не входящий в прототип функции:

Варианты ответов:

1. имя функции
2. адрес возврата в вызывающую программу
3. тип возвращаемого значения
4. типы передаваемых параметров

Критерии оценки выполнения 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

Практическое задание для формирования «ПК-7.2»

РАЗРАБОТКА ШАБЛОННОГО КЛАССА ОЧЕРЕДЬ

Цель работы: разработать шаблонный класс, реализующий процедуру обслуживания FIFO для объектов произвольных типов с основными операциями: добавить в конец очереди; удалить из головы; проверить, не пуста ли очередь; проверить, не заполнена ли очередь.

Пример проектирования.

Наша реализация будет содержать определения двух классов:

1. Класс Queue, предоставляющий интерфейс и члены данные front и back. Очередь реализуем как связный список.

2. Класс QueueItem. Каждый элемент, попадающий в очередь, является объектом этого класса.

Класс QueueItem содержит члены, в которых хранятся значение элемента value и его связь со следующим next элементом. Тип значения может меняться в зависимости от того, каким конкретно является класс Queue.

```
template <class T> class QueueItem {
public:
QueueItem(const T& t);
~QueueItem();
private:
T item;
QueueItem* next;
};
```

Внутри класса имя QueueItem является краткой записью для QueueItem <class T> . Вне класса надо явно указывать список параметров:

```
template <class T> inline QueueItem <T> :: QueueItem(const T& t) :
item(t) {
next = 0;
}
```

Второе вхождение QueueItem в определение конструктора не требует списка формальных параметров шаблона, поскольку в области видимости шаблонного класса оно означает член этого класса и не используется как спецификатор типа.

Теперь можно создавать объекты, которые могут являться элементами очереди:

```
QueueItem<int> qi;
QueueItem<double> qd;
```

Класс QueueItem играет роль вспомогательного для реализации класса Queue и не предназначен для использования во всей программе. Чтобы это гарантировать можно определение его конструктора поместить в private часть класса. Тогда Queue должен быть другом QueueItem причем для каждой конкретизации типа требуется взаимнооднозначное соответствие между Queue и QueueItem. Следовательно, описание должно иметь вид:

```
template <class T> class QueueItem {
friend class Queue<T>;
public:
QueueItem(const T& t);
~QueueItem();
private:
T item;
QueueItem* next;
};
```

Дадим более полное определение класса Queue:

```
enum BOOL {FALSE = 0, TRUE};
//Предварительное описание класса QueueItem
template <class T> class QueueItem;
template <class Type> class Queue {
public:
Queue() {front = back = 0;}
~Queue();
Type remove();
void add(const Type&);
BOOL is_empty() {
return front == 0 ? TRUE : FALSE;
}
private:
QueueItem<Type> * front;
QueueItem<Type> * back;
};
```

Здесь каждая конкретизация Queue приводит к конкретизации соответствующего экземпляра QueueItem.

Реализация функций членов класса может быть следующей.

```
template <class Type> Queue <Type> :: ~Queue() {
while(is_empty() != TRUE)
(void) remove();
}
```

Функция add() добавляет новый элемент в хвост очереди.

```
template <class Type> Queue <Type> :: add(const Type& val) {
QueueItem<Type> * pt =
new QueueItem<Type> (val);
if(is_empty())
front = back = pt;
else
{
back->next = pt;
back = pt;
```

```
}
}
```
Функция remove() выдает элемент из головы очереди производя его удаление:
```
template <class Type> Queue <Type> :: remove() {
if(is_empty() == TRUE) {
cerr << "очередь пуста\n" ;
exit(-1);
}
QueueItem<Type> * pt = front;
front = front->next;
Type retval = pt->item;
delete pt;
return retval;
}
```
Если мы хотим в любой момент существования очереди вывести ее содержимое на экран, придется перегрузить операцию вывода <<. Понятно, что перегрузка операции вывода должна быть шаблонной функцией:
```
template <class Type> ostream& operator << (ostream& , Queue<Type>& );
```
Кроме того, надо описать операцию как друга Queue:
```
template <class Type> class Queue {
friend ostream& operator << (ostream& , Queue<Type>& );
//.....
};
```
Сама же функция может быть реализована следующим образом:
```
template <class Type>
ostream& operator << (ostream& os, Queue<Type>& q){
os << "<";
Queue<Type>* p;
for( p = q.front; p; p = p->next)
os << *p << " ";
os << ">";
return os;
}
```
Задача вывода элемента очереди перекладывается на операцию вывода класа QueueItem:
```
os << *p;
```
Эта операция должна также определяться как шаблонная функция:
```
template <class Type>
ostream& operator << (ostream& os, QueueItem<Type>& qi){
os << q.item;
return os;
}
```
Поскольку в ней используются частные члены QueueItem, она должна быть дружественной этому классу:

```
template <class Type> class QueueItem {
friend class Queue<Type>;
friend ostream& operator << (ostream& , QueueItem<Type>& );
public:
QueueItem(const Type& t);
~QueueItem();
private:
Type item;
QueueItem* next;
};
```

Внимание! Накладываются неочевидные ограничения на конкретизацию Queue. Для каждого типа, для которого образуется очередь, должна быть представлена операция извлечения из потока.

Задание: разработать и отладить класс очередь. Написать программу, иллюстрирующую все определенные в ней операции с объектами класса.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-7.2»

ИЗУЧЕНИЕ ПОТОКОВ ВВОДА-ВЫВОДА

1. Цель работы

Изучить возможности стандартной библиотеки ввода-вывода (iostream.h), манипуляторы (iomanip.h), классы: strstream, ios, fstream.

2. Справочная информация

Традиционные средства ввода-вывода (stdlib.h) были ориентированы на стандартные типы данных. Ввод-вывод данных определяемых пользователем также нуждается в библиотечной поддержке. Соответствующие шаблонные классы включены в состав библиотеки STL.

2.1. Стандартный ввод-вывод

C++ поддерживает четыре предопределенных потоковых объекта:

cin – стандартный ввод (клавиатура)
cout - стандартный вывод (экран)
cerr - стандартный вывод ошибок (экран)

clog – полная буферизированная версия cerr.

Все стандартные потоки можно произвольным образом перенаправить в/из устройства и файлы.

Библиотека классов содержит базовые компоненты (обычно не используемые напрямую):

ios_base
basic_ios
basic_iostream
basic_streambuf

Кроме базовых компонент следует перечислить часто используемые классы и их назначение:

streambuf – буферизация в памяти

ios – состояния потоков и ошибки
istream – форматированное и неформатированное преобразование символов из streambuf
ostream - форматированное и неформатированное преобразование символов в streambuf
iostream – комбинация istream и ostream для поддержки двунаправленных операций

По умолчанию поддерживается вывод объектов следующих типов:

char (signed, unsigned)
short (signed, unsigned)
int (signed, unsigned)
long (signed, unsigned)
char*
float
double
void*

Для вывода двоичных данных и простых символов можно использовать функцию put(), определенную как:

basic_ostream& basic_ostream::put(char_type _Ch);

При этом два следующие оператора эквивалентны:

cout.put(ch);

cout << ch;

Для вывода

объектов определена функция write():

basic_ostream& basic_ostream::write(

const char_type *_Str,

streamsize _Count

);

Эта функция помещает в поток _Count символов в двоичном формате и может использоваться для сериализации объектов:

CAnyClass object;

cout.write(static_cast<char*>(&object),sizeof(CAnyClass));

Если вывод в поток должен сопровождаться форматированием, можно воспользоваться манипуляторами класса ios, которые в свою очередь используют флаги форматирования:

typedef T1 fmtflags;

static const fmtflags boolalpha, dec, fixed, hex,internal, left, oct, right, scientific, showbase, showpoint, showpos, skipws, unitbuf,uppercase, adjustfield, basefield, floatfield;

Тип T1 соответствует битовой маске, определяющей следующие флаги форматирования:

dec, вставка или выделение целых величин в десятичном формате.
hex, вставка или выделение целых величин в шестнадцатиричном формате.
oct, вставка или выделение целых величин в восмеричном формате.

showbase, вставка префикса основания системы счисления для целых.
internal, вставка символов заполнения, до получения заданной ширины поля целого типа, после знака или основания системы счисления.
left, вставка символов заполнения (левое выравнивание).
right, вставка символов заполнения (правое выравнивание).
boolalpha, вставка или выделение объекта типа bool как имени (т.е. как true и false) в противовес числовому представлению.
fixed, вставка чисел с плавающей точкой в формате с фиксированной точкой.
scientific, вставка чисел с плавающей точкой в научном формате.
showpoint, отображать десятичную точку при всавке чисел с плавающей точкой.
showpos, вставлять знак + для неотрицательных числовых полей.
skipws, исключить лидирующие пробелы.
unitbuf, очистка буфера после каждой вставки.
uppercase, вставить эквивалентный символ верхнего регистра.

Перечисленные флаги могут использоваться как объединение по битовой операции “ИЛИ”. Наиболее часто используемые комбинации соответствуют приведенным ниже константам:

adjustfield = internal | left | right;
basefield = dec | hex | oct;
floatfield = fixed | scientific;

Проверить и изменить состояние флагов можно с помощью функции flags():

```
using namespace std;

cout << cout.flags( ) << endl;

cout.flags( ios::dec | ios::boolalpha );

cout << cout.flags( );
```

Простой и удобный способ установки формата заключается в использовании манипуляторов, представляющих собой функции класса ios_base меняющие состояние флагов формата и возвращающие ссылку на объект ios_base, благодоря чему манипуляторы могут конкатенироваться с операциями вставки в поток. Наряду с манипуляторами:

resetiosflags – сбрасывает специфицированный флаг
setbase – устанавливает основание системы счисления для целых.
setfill – устанавливает символы заполнения.
setiosflags - устанавливает специфицированные флаги.
setprecision – устанавливает точность для чисел с плавающей точкой.
setw – устанавливает ширину поля.

Можно пользоваться функциями, ведущими себя как манипуляторы:

dec – вставка или выделение целых величин в десятичном формате
hex – вставка или выделение целых величин в шестнадцатиричном формате
oct – вставка или выделение целых величин в восмеричном формате
ws – удаление пробелов
endl – перевод строки
ends – вставка 0 символа

Пример использования манипуляторов приведен ниже:

```
#include <iostream>

int main( )

{

using namespace std;

int i = 100;

// По умолчанию основание системы счисления 10

cout << i << endl;
```

```
cout << hex << i << endl;
dec( cout );
cout << i << endl;
oct( cout );
cout << i << endl;
cout << dec << i << endl;
}
```

2.2. Инициализация потоков

Потоки cin, cout, cerr, clog инициализируются при запуске программы в стартовом коде CRT библиотеки выполняемом до вызова функции main(). Инициализация потока есть связывание его с буфером потока. Такое связывание обеспечивается при конструировании объекта basic_ios

```
explicit basic_ios(
basic_streambuf<_Elem,
_Traits> *_Sb
);
```

Конструктор инициализирует флаги состояния basic_ios и ассоциирует буфер потока с объектом basic_ios.

Из абстрактного класса basic_streambuf порождены:

- basic_filebuf – поддерживает файловый ввод-вывод
- basic_stringbuf – ввод-вывод в память

Для файловых операций ввода-вывода используются специализации:

- ifstream – файловый поток ввода
- ofstream – файловый поток вывода
- fstream – файловый поток ввода-вывода

Ввод-вывод в памяти может быть организован с помощью специализаций:

- stringstream
- wstringstream

2.3. Ввод-вывод для файлов

Продемонстрировать файловый ввод-вывод можно следующим образом:

```
#include <fstream>
using namespace std;
char ch;
ifstream inFile(_T("readMe.txt"));
if(inFile.is_open()){
ofstream outFile(_T("writeMe.txt"));
if(outFile.is_open()){
while(inFile&&inFile.get(ch))
outFile.put(ch);
}
else{
cerr<<_T("Не могу открыть файл для записи")<<endl;
}
}
else{
```

```
cerr<<_T("Не могу открыть файл для чтения")<<endl;
}
```

В приведенном примере осуществляется копирование содержимого входного файла в выходной. Привязка к файлу производится при конструировании объекта. Однако привязка может выполняться и после вызова конструктора:

```
ofstream ofs;
. . . . . .
ofs.open(_T("writeMe.txt"));
…….
ofs.close();
ofs.open(_T("w.txt"));
```

По умолчанию файлы открываются в текстовом режиме, однако можно изменить режим открытия, задав необходимые аргументы конструктора потока или функции open():

```
explicit basic_fstream(
const char *_Filename,
ios_base::openmode _Mode = ios_base::in | ios_base::out,
int _Prot = (int)ios_base::_Openprot
);
explicit basic_fstream(
const wchar_t *_Filename,
ios_base::openmode _Mode = ios_base::in | ios_base::out,
int _Prot = (int)ios_base::_Openprot
);
void open(
const char *_Filename,
ios_base::openmode _Mode = ios_base::in | ios_base::out,
int _Prot = (int)ios_base::_Openprot
);
void open(
const char *_Filename,
ios_base::openmode _Mode
);
void open(
const wchar_t *_Filename,
ios_base::openmode _Mode = ios_base::in | ios_base::out,
int _Prot = (int)ios_base::_Openprot
);
void open(
const wchar_t *_Filename,
ios_base::openmode _Mode
);
```

Режим открытия определен следующим образом:

```
static const _Openmode in = (_Openmode)0x01;
```

static const _Openmode out = (_Openmode)0x02;

static const _Openmode ate = (_Openmode)0x04;

static const _Openmode app = (_Openmode)0x08;

static const _Openmode trunc = (_Openmode)0x10;

static const _Openmode _Nocreate = (_Openmode)_IOS_Nocreate;

static const _Openmode _Noreplace = (_Openmode)_IOS_Noreplace;

static const _Openmode binary = (_Openmode)_IOSbinary;

Перечисленные значения представляют собой битовую маску и могут комбинироваться с помощью битовой операции ‘ИЛИ’:

app, to seek to the end of a stream before each insertion.
ate, to seek to the end of a stream when its controlling object is first created.
binary, to read a file as a binary stream, rather than as a text stream.
in, to permit extraction from a stream.
out, to permit insertion to a stream.
trunc, to delete contents of an existing file when its controlling object is created.

Аргумент _Prot определяет защиту по умолчанию. Если не задавать этот аргумент, то он примет значение, определенное в перечислителе:

enum{ // constant for default file opening protection

_Openprot = _SH_DENYNO};

};

Можно задать значения аргумента _Prot из перечисленных в заголовочном файле share.h:

#define _SH_DENYRW 0x10 /* deny read/write mode */

#define _SH_DENYWR 0x20 /* deny write mode */

#define _SH_DENYRD 0x30 /* deny read mode */

#define _SH_DENYNO 0x40 /* deny none mode */

#define _SH_SECURE 0x80 /* secure mode */

Для потоков требующих как ввод в файл, так и вывод из файла можно воспользоваться fstream:

fstream io(_T("data.txt"),ios::in|ios::out);

io<<ch;

io>>ch;

io.close();

Текущая позиция ввода и вывода может быть определена функциями tellg(), tellp(), а установка нужной позиции в файле функциями seekg(), seekp().

2.4. Ошибочные состояния

Каждый поток имеет связанные с ним флаги состояния с помощью которых может осуществляться контроль ошибок.

Соответствующие состояниям биты определены в заголовочном файле xiobase.h:

static const _Iostate goodbit = (_Iostate)0x0;

static const _Iostate eofbit = (_Iostate)0x1;

static const _Iostate failbit = (_Iostate)0x2;

static const _Iostate badbit = (_Iostate)0x4;

static const _Iostate _Hardfail = (_Iostate)0x10;

Функция void basic_ios::clear(iostate _State=goodbit); устанавливает биты состояния в соответствии с аргументом _State. Вызов этой функции с аргументом по умолчанию сбрасывает все биты ошибок за исключением _Hardfail. Для проверки битов ошибки и состояния потока можно воспользоваться функциями ios_base:

iostate ios_base::rdstate() const;

bool ios_base::eof() const;

bool ios_base::fail() const;

bool ios_base::bad() const;

bool ios_base::good() const;

Если поток находится в состоянии good, значит предыдущие операции выполнены успешно. Различие между fail и bad очень незначительно. Когда поток в состоянии fail, но не bad, считается что он не испорчен и потери символов не произошло. Если поток в состоянии bad, то ни в чем нельзя быть уверенным. Благодоря определению операций operator void *( ) const; и bool operator!( ) const; возможен контроль ошибок в потоках использующий логические выражения:

```
if(cin>>x){
……..
}
else cerr<<_T(("Ошибка ввода!");
```

3. Задание

Написать приложение с использованием потоков ввода-вывода для сохранения в файле структурированной информации о студенте. Обеспечить операции чтения, вставки, редактирования, поиска записей.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Практическое задание для формирования «ПК-7.3»

Практические задания

Разработать модель системы, построенную на архитектуре "клиент/сервер", автоматизирующую работу инфокоммуникационной системы учебного центра.

Методологической основой разработки является унифицированный процесс RUP. Каждое практическое занятие посвящено определенным фазам и основным потокам работ.

Будут разработаны следующие диаграммы:

вид с точки зрения функциональности – диаграммы прецедентов;

вид с точки зрения проектирования – диаграммы классов (для структурного моделирования) и диаграммы взаимодействия (для моделирования поведения).

Практическое занятие 1. Спецификация требований и создание вариантов использованияЦель

Уяснить принципы документирования требований к ПО.

Исходные данные

Учебный центр Program System проводит обучение по официальным программам всемирно известной компании Macro. На начальном этапе своего развития учебный центр начинался с небольшой группы собравшихся вместе преподавателей. В то время компания Macro еще не доминировала на мировом рынке ПО, объем программ обучения был не велик и с ним успешно справлялся коллектив учебного центра. С ростом известности и компании Macro и репутации учебного центра, объем заявок на обучение возрастал. Когда поступало мало заявок, менеджер центра использовал для их учета бумажный журнал, а в случае крайней необходимости простые электронные таблицы Excel.

Проблемы

В последнее время все чаще клиенты стали жаловаться, что не могут оформить заявку на обучение, причем по разным причинам, например, они не могут быстро выбрать подходящий курс обучения (информации на сайте не достаточно для правильного выбора), или в силу занятости менеджера оформление проходит слишком долго и клиенты выбирают другие центры.

При регистрации на обучение клиенты заполняют анкету и передают ее в секретариат. В случае возникновения конфликтов (например, нет мест, перенос даты занятий, смена места обучения) клиентам сообщается о возможных изменениях. Таким образом, процесс регистрации затягивается, и нетерпеливые клиенты выбирают другие центры.

Если так будет продолжаться и дальше, ученый центр может потерять свои лидирующие позиции.

Решение проблемы

Разработка системы, которая поможет автоматизировать процесс обработки заявок клиентов.

Описание процесса регистрации клиентов на обучение

Клиенты должны иметь возможность просмотреть каталог доступных курсов в данный период. В него включена информация о каждом курсе, имя преподавателя и предварительные требования, необходимые для качественного усвоения материала.

Ожидается, что новая автоматизированная система позволит клиентам выбирать курсы и, по возможности, преподавателя. Дополнительно каждому клиенту должен быть предложен альтернативный вариант в случае, если на курсе нет мест или он отменен, а также перенесен. На одном курсе не может быть больше 25 и меньше пяти обучаемых. Курсы, на которые записались менее пяти клиентов, переносятся на неделю, при этом необходимо учесть возможную занятость преподавателя на другом курсе. Как только группа обучения сформирована, процесс регистрации клиентов завершается, и регистрационная система отсылает информацию билинговой системе для выписки клиентам счетов за обучение.

Преподаватели могут получить доступ к информации о своих курсах и зарегистрированных клиентах, например, посмотреть опыт их работы и уровень подготовленности, определенный по результатам входного тестирования.

Менеджер формирует набор курсов, распределяет преподавателей и оформляет клиентов. Администратор (лаборант) обеспечивает подготовку учебных классов для проведения курса.

Администратор центра тестирования организует работу центра для приема сертифицированных экзаменов.

Руководитель центра обучения организует работу всего центра.

В целях повышения квалификации преподавателей центра им разрешается посещать курсы других преподавателей. В этом случае необходимо пройти регистрацию в качестве обучаемого, но обучение будет для них бесплатно.

Реализация начальной фазы

На основе исходных данных (пожелания и требования заказчика по функционированию системы) требуется:

Сформировать концепцию – образ проекта в целом. Предварительно оценить возможные риски и необходимые ресурсы.

Составить план, в котором отразить основные опорные точки процесса разработки. Определить основную функциональность, которую должна предоставлять система.

На основе функциональных требований создать модель прецедентов (вариантов использования).

Рекомендации по разработке:

Данная модель показывает функции системы (собственно варианты использования), их окружение (актеры) и связи (отношения) между прецедентами и актерами.

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует следующие цели:

определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.

сформулировать общие требования к функциональному поведению проектируемой системы. разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчикамии пользователями.

Отчетность

Отчет должен содержать:

описание исходных данных проекта;план разработки системы;

спецификация требований к ПО согласно шаблонадиаграмму вариантов использования;

На диаграммах вариантов использования каждое действующее лицо и вариант использования должны сопровождаться описанием. Описание действующего лица должно коротко (в одну-две строки) сообщать о роли данного лица. Описание варианта использования должно включать в себя пояснение, предусловие, потоки событий (основной и альтернативные, если таковые есть) и постусловие.

выводы;

В выводах необходимо определить, правильно ли реализована фаза «Начало», в этом случае она должна устанавливать высокоуровневые требования для желаемой и осуществимой системы.

Неадекватная фаза делает систему далекой от желаемой, плохо описанной и слишком дорогой, в результате чего она никогда не будет реализована.

Практическое занятие 2. Разработка диаграммы взаимодействияЦель

Уяснить принципы разработки диаграмм взаимодействия, соответствующие потокам событий вариантов использования.

Порядок выполнения

Для полученных в предыдущей работе вариантов использования требуется описать сценарий использования с помощью диаграммы последовательности.

Рекомендации по разработке

Диаграмма взаимодействий акцентирует внимание на временной упорядоченности сообщений. Графически такая диаграмма представляет собой таблицу, объекты в которой располагаются вдоль оси X, а сообщения в порядке возрастания времени – вдоль оси Y.

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования, и демонстрирует взаимодействие объектов, упорядоченное по времени.

Объект является представлением сущности либо из реального мира, либо концептуальной модели. Объект может представлять нечто конкретное, например, грузовик или компьютер, или концепцию, например химический процесс, банковскую транзакцию, чек на покупку.

Класс описывает группу объектов с общими свойствами (атрибутами), общим поведением

(операциями), общими связями с другими объектами и общей семантикой (шаблон для создания объекта).

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами.

Для диаграммы последовательности ключевым моментом является именно динамика

взаимодействияобъектов во времени.

При этом диаграмма последовательности имеет как бы два измерения. Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии.

Второе измерение диаграммы последовательности – вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим.

Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни. Внутри прямоугольника записываются имя объекта и имя класса, разделенные двоеточием. При этом вся запись подчеркивается, что является признаком объекта, который, как известно, представляет собой экземпляр класса.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия

Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют некоторый порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Отчетность

Отчет должен содержать:

диаграммы взаимодействия, соответствующие потокам событий вариантов использования с соответствующими пояснениями;

выводы.

Практическое занятие 3. Реализация иерархии классовЦель

Разработать диаграмму классов и программно реализовать иерархию классов.Порядок выполнения

Работа состоит из двух частей.

В первой части требуется разработать диаграмму классов. Для каждого класса продумать атрибуты,конструкторы и методы.

Обратите особое внимание на отношения между классами, подпишите тип отношения (является, содержит или реализует) над каждой стрелкой на диаграмме.

Реализуйте, где это необходимо, механизм композиции. В этом случае поведение не наследуется, а предоставляется правильно выбранным объектом.

Во второй части работы требуется реализовать диаграмму классов на языке C#. Протестировать работуклассов.

Рекомендации по разработке

Для декомпозиции системы на классы следует выделить следующие три стереотипа классов: граничные классы, управляющие классы (контроллеры) и классы-сущности, которые образуют концепцию «модель-представление-контроллер» (MVC) и позволяют отделить представление от предметной области и управления.

Классы-сущности отражают сущности реального мира или выполняют внутренние задачи системы. Эти классы необходимы системе для выполнения определенных функций. Обычно они не зависят от

того, как система общается с внешним окружением, и могут быть использованы в нескольких приложениях.

Граничные классы обрабатывают взаимодействие между окружением системы и ее внутренними частями. Они могут представлять интерфейс пользователям (см. следующую работу) или другим системам. В этой работе требуется сделать прототипы (наброски) классов для отображения интерфейса, которые в дальнейшем будут уточнены.

Управляющие классы отражают поведение одного или нескольких прецедентов использования и отвечают за поток сообщений в прецеденте использования. Эти классы зависят от конкретного приложения.

Отчетность

Отчет должен содержать:диаграммы классов;
код, реализующий иерархию классов;выводы.

Убедитесь, что системы, созданные на основе композиции, обладают значительно большей гибкостью, они позволяют не только инкапсулировать семейства алгоритмов, но и изменять поведение во время выполнения, при условии, что объект, подключенный посредством композиции, реализует правильныйинтерфейс.

Критерии оценки выполнения задания

Оценка	Критерии оценивания
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

Вопросы для проведения промежуточной аттестации по итогам освоения дисциплины

Тема 1. Программное средство как продукт технологии программирования

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.

Тема 2. Основные этапы процесса проектирования программного обеспечения

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.

Тема 4. Методы проектирования и разработки программного обеспечения

18. Понятие методологии.
19. Атрибуты методологий.
20. Основные методологии программирования.

Тема 5. Методология объектно-ориентированного 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.

Тема 8. Технологические средства разработки программного обеспечения

34. Языки программирования четвертого поколения, CASE-системы, системы ускоренной разработки 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, взаимодействия объектов, 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 программного обеспечения.
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.

Тема 10. Методы отладки и тестирования программ

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.

Уровни и критерии итоговой оценки результатов освоения дисциплины

	Критерии оценивания	Итоговая оценка

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