МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего образования
«Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

ОТДЕЛЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Одобрено на заседании Ученого совета ИАТЭ НИЯУ МИФИ, протокол №25.8 от 25.08.2025

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по дисциплине

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

название дисциплины

для направления подготовки

38.03.05 Бизнес-информатика

образовательная программа

Цифровая трансформация компании

код и название направления подготовки

Форма обучения: очная

г. Обнинск 2025 г.

Область применения

Фонд оценочных средств (ФОС) – является неотъемлемой частью учебно-методического комплекса учебной дисциплины «Объектно-ориентированное программирование» и предназначен для контроля и оценки образовательных достижений обучающихся, освоивших программу данной дисциплины.

Цели и задачи фонда оценочных средств

Целью Фонда оценочных средств является установление соответствия уровня подготовки обучающихся требованиям федерального государственного образовательного стандарта.

Для достижения поставленной цели Фондом оценочных средств по дисциплине «Объектно-ориентированное программирование» решаются следующие задачи:

– контроль и управление процессом приобретения обучающимися знаний, умений и навыков предусмотренных в рамках данного курса;

– контроль и оценка степени освоения компетенций предусмотренных в рамках данного курса;

– обеспечение соответствия результатов обучения задачам будущей профессиональной деятельности через совершенствование традиционных и внедрение инновационных методов обучения в образовательный процесс в рамках данного курса.

Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

В результате освоения ООП бакалавриата обучающийся должен овладеть следующими результатами обучения по дисциплине:

Коды компетенций	Наименование компетенции	Код и наименование индикатора достижения компетенции
ОПК-3	Способен управлять процессами создания и использования продуктов и услуг в сфере ИКТ, в том числе разрабатывать алгоритмы и программы для их практической реализации	З-ОПК-3 Знать: Методы оценки объемов и сроков выполнения работ Технологии выполнения работ в организации Технологии межличностной и групповой коммуникации в деловом взаимодействии, основы конфликтологии Архитектура, устройство и функционирование вычислительных систем Коммуникационное оборудование Сетевые протоколы Основы современных операционных систем Основы современных систем управления базами данных Устройство и функционирование современных ИС Теория баз данных Системы хранения и анализа баз данных

Коды компетенций	Наименование компетенции	Код и наименование индикатора достижения компетенции
		Основы программирования Современные объектно-ориентированные языки программирования Современные структурные языки программирования Языки современных бизнес-приложений Современные методики тестирования разрабатываемых ИС Современные стандарты информационного взаимодействия систем Программные средства и платформы инфраструктуры информационных технологий организаций Современные подходы и стандарты автоматизации организации (например, Web-системы, CRM, MRP, ERP..., ITIL, ITSM) Системы классификации и кодирования информации, в том числе присвоение кодов документам и элементам справочников Отраслевая нормативная техническая документация Источники информации, необходимой для профессиональной деятельности У-ОПК-3 Уметь: Разрабатывать документы Оценивать объемы работ и сроки их выполнения Проводить переговоры В-ОПК-3 Владеть навыками: Подготовка частей коммерческого предложения заказчику касательно объема и сроков выполнения работ по созданию (модификации) и вводу в эксплуатацию типовой ИС Осуществление инженерно-технологической поддержки в ходе согласования коммерческого предложения с заказчиком

Этапы формирования компетенций в процессе освоения ООП бакалавриата

Компоненты компетенций, как правило, формируются при изучении нескольких дисциплин, а также в немалой степени в процессе прохождения практик, НИР и во время самостоятельной работы обучающегося. Выполнение и защита ВКР являются видом учебной деятельности, который завершает процесс формирования компетенций.

Место дисциплины и соответствующий этап формирования компетенций в целостном процессе подготовки по образовательной программе можно определить по матрице компетенций, которая приводится в Приложении.

Этапы формирования компетенции в процессе освоения дисциплины:

- **начальный** этап – на этом этапе формируются знаниевые и инструментальные основы компетенции, осваиваются основные категории, формируются базовые умения. Студент воспроизводит термины, факты, методы, понятия, принципы и правила; решает учебные задачи по образцу;
- **основной** этап – знания, умения, навыки, обеспечивающие формирование компетенции, значительно возрастают, но еще не достигают итоговых значений. На этом этапе студент осваивает аналитические действия с предметными знаниями по дисциплине, способен самостоятельно решать учебные задачи, внося коррективы в алгоритм действий, осуществляя коррекцию в ходе работы, переносит знания и умения на новые условия;
- **завершающий** этап – на этом этапе студент достигает итоговых показателей по заявленной компетенции, то есть осваивает весь необходимый объем знаний, овладевает всеми умениями и навыками в сфере заявленной компетенции. Он способен использовать эти знания, умения, навыки при решении задач повышенной сложности и в нестандартных условиях.

Этапы формирования компетенций в ходе освоения дисциплины отражаются в тематическом плане (см.п. 4 рабочей программы дисциплины).

Паспорт фонда оценочных средств по дисциплине «Объектно-ориентированное программирование»

№ п/п	Контролируемые модули, разделы дисциплины	Код контролируемой компетенции	Наименование оценочного средства
1	Раздел 1. Введение в программирование на C++	З-ОПК-3; У-ОПК-3; В-ОПК-3	Письменно. Лабораторная работа №1. Письменно.
2	Раздел 2. Основы объектно-ориентированного программирования	З-ОПК-3; У-ОПК-3; В-ОПК-3	Контрольная работа №1. Лабораторная работа №2. Письменно.
3	Раздел 3. Основы объектно-ориентированного проектирования	З-ОПК-3; У-ОПК-3; В-ОПК-3	Контрольная работа №2. Лабораторная работа №3. Письменно.
4	Раздел 4. Введение в программирование на C# и Java	З-ОПК-3; У-ОПК-3; В-ОПК-3	Лабораторная работа №4. Письменно.

Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Конечными результатами освоения программы дисциплины являются сформированные когнитивные дескрипторы «знать», «уметь», «владеть», расписанные по отдельным компетенциям, которые приведены в п.1.1. Формирование этих дескрипторов происходит в процессе изучения дисциплины по этапам в рамках различного вида учебных занятий и самостоятельной работы.

Выделяются три уровня сформированности компетенций на каждом этапе: пороговый, продвинутый и высокий.

Уровни	Содержательное описание уровня	Основные признаки выделения уровня	БРС, % освоения	ECTS/Пятибалльная шкала для оценки экзамена/зачета
Высокий *Все виды компетенций сформированы на высоком уровне в соответствии с целями и задачами дисциплины*	Творческая деятельность	*Включает нижестоящий уровень.* Студент демонстрирует свободное обладание компетенциями, способен применить их в нестандартных ситуациях: показывает умение самостоятельно принимать решение, решать проблему/задачу теоретического или прикладного характера на основе изученных методов, приемов, технологий	90-100	A/ Отлично/ Зачтено
Продвинутый *Все виды компетенций сформированы на продвинутом уровне в соответствии с целями и задачами дисциплины*	Применение знаний и умений в более широких контекстах учебной и профессиональной деятельности, нежели по образцу, большей долей самостоятельности и инициативы	*Включает нижестоящий уровень.* Студент может доказать владение компетенциями: демонстрирует способность собирать, систематизировать, анализировать и грамотно использовать информацию из самостоятельно найденных теоретических источников и иллюстрировать ими теоретические положения или обосновывать практику применения.	85-89	B/ Очень хорошо/ Зачтено
			75-84	C/ Хорошо/ Зачтено
Пороговый *Все виды компетенций сформированы на пороговом уровне*	Репродуктивная деятельность	Студент демонстрирует владение компетенциями в стандартных ситуациях: излагает в пределах задач курса теоретически и практически контролируемый материал.	65-74	D/Удовлетворительно / Зачтено
			60-64	E/Посредственно /Зачтено
Ниже порогового	Отсутствие признаков порогового уровня: компетенции не сформированы. Студент не в состоянии продемонстрировать обладание компетенциями в стандартных ситуациях.		0-59	Неудовлетворительно / Зачтено

Оценивание результатов обучения студентов по дисциплине осуществляется по регламенту текущего контроля и промежуточной аттестации.

Критерии оценивания компетенций на каждом этапе изучения дисциплины для каждого вида оценочного средства и приводятся в п. 4 ФОС. Итоговый уровень сформированности компетенции при изучении дисциплины определяется по таблице. При этом следует понимать, что граница между уровнями для конкретных результатов освоения образовательной программы может смещаться.

Уровень сформированности компетенции	**Текущий контроль**	**Промежуточная аттестация**
высокий	**высокий**	**высокий**
	продвинутый	*высокий*
	высокий	*продвинутый*
продвинутый	*пороговый*	*высокий*
	высокий	*пороговый*
	продвинутый	**продвинутый**
	продвинутый	*пороговый*
	пороговый	*продвинутый*
пороговый	**пороговый**	**пороговый**
ниже порогового	**пороговый**	**ниже порогового**
	ниже порогового	**-**

Методические материалы, определяющие процедуры оценивания знаний, умений, навыков или опыта деятельности, характеризующих этапы формирования компетенций.

Рейтинговая оценка знаний является интегральным показателем качества теоретических и практических знаний и навыков студентов по дисциплине и складывается из оценок, полученных в ходе текущего контроля и промежуточной аттестации.

Текущий контроль в семестре проводится с целью обеспечения своевременной обратной связи, для коррекции обучения, активизации самостоятельной работы студентов.

Промежуточная аттестация предназначена для объективного подтверждения и оценивания достигнутых результатов обучения после завершения изучения дисциплины.

Текущий контроль осуществляется два раза в семестр: контрольная точка № 1 (КТ № 1) и контрольная точка № 2 (КТ № 2).

Результаты текущего контроля и промежуточной аттестации подводятся по шкале балльно-рейтинговой системы.

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков. Экзаменационные билеты

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»

Обнинский институт атомной энергетики –

филиал федерального государственного автономного образовательного учреждения высшего образования
«Национальный исследовательский ядерный университет «МИФИ»

(ИАТЭ НИЯУ МИФИ)

Отделение Интеллектуальных кибернетических систем

Направление/ Специальность	**380305 «Бизнес-информатика»**
Образовательная программа	**«Цифровая трансформация компании»**
Дисциплина	**«Объектно-ориентированное программирование»**

Экзаменационный билет № 1

Вопрос №1

Основные операции языка Си.

Вопрос №2

Язык Си: низкоуровневый ввод/вывод. Пример.

Вопрос № 3

Модели жизненного цикла программного обеспечения.

Начальник ОИКС _____ Старков С.О.

Критерии оценки:

Оценка,%	Характеристики ответа студента
76 – 100%	– знания отличаются глубиной и содержательностью, даны логично построенные, полные, исчерпывающие ответы на вопросы; – обучающийся демонстрирует способность к анализу практических ситуаций, оперирует профессиональными терминами; – обучающийся умеет высказывать и обосновать свои суждения;
51 – 75%	– знания имеют достаточный содержательный уровень, однако имеются

	определенные затруднения в ответе на уточняющие вопросы; – несущественные ошибки в определении понятий и категорий и т. п., кардинально не меняющие суть изложения;
26 – 50%	– знания имеют фрагментарный характер, имеются определенные неточности и погрешности в формулировках, возникают затруднения при ответе на уточняющие вопросы; – при ответе на вопросы обучающийся не может обосновать закономерности, принципы, объяснить суть явления; допущены фактические ошибки; – обучающийся продемонстрировал слабое умение формулировать выводы и обобщения,
0 – 25%	– обучающийся слабо ориентируется в теме задания, затрудняется дать ответ или дает неверные ответы на поставленные вопросы; – допущены существенные фактические ошибки при ответах на вопросы.

Прочие экзаменационные билеты

Экзаменационный билет № 2

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: переключатели, перечислимые типы. Примеры.

Вопрос № 2

Язык Си: примеры реализаций функций ввода/вывода (getc, putc).

Вопрос № 3

Проектирование программного обеспечения и UML.

Экзаменационный билет № 3

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: указатели и массивы. Определения, примеры.

Вопрос № 2

Язык Си: примеры реализаций функций ввода/вывода (fgets, fputs).

Вопрос № 3

Диаграммы прецедентов. Нотация, семантика, примеры.

Экзаменационный билет № 4

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: объявления функций, передача аргументов. Примеры.

Вопрос № 2

Язык Си: работа с файловой системой. Пример.

Вопрос № 3

Сценарии выполнения прецедентов (пример).

Экзаменационный билет № 5

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: строки и указатели. Определения, примеры.

Вопрос № 2

Основные понятия ООП: абстракция, инкапсуляция, наследование, полиморфизм.

Вопрос № 3

Диаграммы классов. Нотация, семантика, отношения.

Экзаменационный билет № 6

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: аргументы командной строки. Пример.

Вопрос № 2

Альтернативное определение ООП.

Вопрос № 3

Атрибуты на диаграммах классов. Нотация и семантика. Примеры.

Экзаменационный билет № 7

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: указатели на функции. Пример.

Вопрос № 2

Объекты классов: статические, автоматические, динамические. Примеры.

Вопрос № 3

Операции на диаграммах классов. Нотация и семантика. Примеры.

Экзаменационный билет № 8

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: структуры. Пример.

Вопрос № 2

Управление доступом к элементам классов. Пример.

Вопрос № 3

Отношения ассоциации на диаграммах классов.

Экзаменационный билет № 9

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: битовые поля и объединения. Примеры.

Вопрос № 2

Шаблоны функций и шаблоны классов. Примеры.

Вопрос № 3

Отношения обобщения на диаграммах классов.

Экзаменационный билет № 10

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: оператор определения типа. Примеры.

Вопрос № 2

Наследование. Пример.

Вопрос № 3

Отношения агрегации на диаграммах классов.

Экзаменационный билет № 11

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: препроцессор. Директивы препроцессора, примеры.

Вопрос № 2

Множественное наследование. Пример.

Вопрос № 3

Отношения композиции на диаграммах классов.

Экзаменационный билет № 12

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: программный стек. Пример работы стека.

Вопрос № 2

Виртуальные функции. Раннее и позднее связывание.

Вопрос № 3

Обзор платформы .NET Framework. C# как язык ООП для .NET Framework.

Экзаменационный билет № 13

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Линейные списки. Операции с линейными списками.

Вопрос № 2

Абстрактные классы. Их назначение, пример.

Вопрос № 3

Язык C#: сборки, манифесты, домены, компоненты.

Экзаменационный билет № 14

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Hash-таблицы (таблицы с перемешиванием).

Вопрос № 2

Полиморфный контейнер (пример).

Вопрос № 3

Обзор различий между языками C# и C++.

Экзаменационный билет № 15

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Двоичные деревья. Алгоритмы обхода, поиска, вставки, сортировки.

Вопрос № 2

Виртуальная схема наследования.

Вопрос № 3

Java как язык ООП для машинно-назависимых приложений.

Экзаменационный билет № 16

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: перегруженные функции. Пример.

Вопрос № 2

Конструкторы, их виды, примеры. Вызов конструкторов при наследовании.

Вопрос № 3

Обзор Java-технологий NetBeans IDE.

Экзаменационный билет № 17

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: ссылочные типы. Пример.

Вопрос № 2

Деструкторы. Их назначение, примеры.

Вопрос № 3

Компонентные технологии разработки программного обеспечения.

Экзаменационный билет № 18

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: защита указателей и объектов, неявное изменение объектов.

Вопрос № 2

Перегруженные операции. Формализм на языке C++.

Вопрос № 3

Обзор различий между языками Java и C++.

Экзаменационный билет № 19

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: организация ввода/вывода. Пример: слияние файлов.

Вопрос № 2

Перегруженные операции. Примеры.

Вопрос № 3

Обёртки в языках C# и Java.

Экзаменационный билет № 20

по курсу «Объектно-ориентированное программирование» для специальности 230100

Вопрос № 1

Язык Си: произвольный доступ к файлам. Пример.

Вопрос № 2

Обработка нештатных ситуаций. Объекты-исключения. Примеры.

Вопрос № 3

Архитектурный паттерн MVC. Область применения, схема взаимодействия.

Перечень оценочных средств

№ п/п	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
1	2	3	4
1	Контрольная работа	Средство проверки умений применять полученные знания для решения задач определенного типа по теме или разделу	Комплект контрольных заданий по вариантам
2	Проект	Конечный продукт, получаемый в результате планирования и выполнения комплекса учебных и исследовательских заданий. Позволяет оценить умения обучающихся самостоятельно конструировать свои знания в процессе решения практических задач и проблем, ориентироваться в информационном пространстве и уровень сформированности аналитических, исследовательских навыков, навыков практического и творческого мышления. Может выполняться в индивидуальном порядке или группой обучающихся.	Темы групповых и/или индивидуальных проектов в форме заданий лабораторного практикума

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский ядерный университет «МИФИ»

Обнинский институт атомной энергетики –

филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»

(ИАТЭ НИЯУ МИФИ)

Комплект заданий для контрольной работы

по дисциплине «Объектно-ориентированное программирование»

Контрольная работа №1

Вариант 1

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
char*(*(*q)(float*));
```

2. Разработайте макрос **swap(x,y,t)** который меняет местами значения **x** и **y**. Тип **x** и **y** есть **t**.
3. Разработайте функцию, которая размещает содержимое дискового файла в динамической памяти. Имя файла задано в командной строке.

Вариант 2

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
double*(*(*p)(void*))(void);
```

2. Разработайте макрос **swab(x)** который меняет местами левый и правый байты целого аргумента **x.**
3. Разработайте функцию, которая находит первое появление заданного символа в дисковом файле. Имя файла задано в командной строке.

Вариант 3

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
union { char* a, b; float x[9]; } z;
```

2. Разработайте макрос **isalpharus(c)** который проверяет, принадлежит ли символ **c** множеству русских букв.
3. Разработайте функцию, которая выводит содержимое дискового файла на печать. Имя файла задано в командной строке.

Вариант 4

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
union z { long* a; int* b; float* c, d; };
```

2. Разработайте макрос **isdigit(c)** который проверяет, принадлежит ли символ **c** множеству цифр.
3. Разработайте функцию, которая сравнивает две строки, считая символы верхнего и нижнего регистра эквивалентными.

Вариант 5

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
float*(*(*f)(void*))[];
```

2. Разработайте макрос **isspace(c)** который проверяет, принадлежит ли символ **c** множеству пробельных символов.
3. Разработайте функцию, которая попеременно читает по одной строке сначала из первого файла, затем из второго и все прочитанное помещает в стандартный вывод в указанном порядке. Имена файлов заданы в командной строке.

Вариант 6

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
struct z { long a,*b; struct z*x };
```

2. Разработайте макрос **buffer(n)** который резервирует в памяти буфер емкостью **n** байтов и при каждом обращении выдает адрес очередного байта или **NULL**, если буфер исчерпан.
3. Разработайте функцию, которая склеивает две строки в одну. Используйте динамическую память.

Вариант 7

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
struct z *(*p[7])();
```

2. Разработайте макрос **minus(x)** который делает **x** отрицательным.
3. Разработайте программу, которая «смешивает» произвольное количество дисковых файлов следующим образом: попеременно читает по одному символу из каждого файла и все прочитанное помещает в стандартный вывод. Имена файлов заданы в командной строке.

Вариант 8

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
double (*s(double,void*))[3];
```

2. Разработайте макрос **itoal(x)** который возвращает символ, расположенный в левом байте целого **x**.
3. Разработайте функцию, которая выводит на печать **n** первых аргументов командной строки. Предусмотрите обработку некорректных значений **n.**

Вариант 9

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
void (*p)(void*,char,...);
```

2. Разработайте макрос **itoar(x)** который возвращает символ, расположенный в правом байте целого **x**.
3. Разработайте функцию, которая выводит на печать **n** последних аргументов командной строки. Предусмотрите обработку некорректных значений **n.**

Вариант 10

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
struct { char*(*p)(void),d; } a[256];
```

2. Разработайте макрос **ltoal(x)** который возвращает символ, расположенный в самом левом байте длинного целого **x**.
3. Разработайте функцию, которая добавляет **n** заданных символов в строку, начиная с указанной позиции. Исходная строка при этом раздвигается. Используйте динамическую память.

Вариант 11

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
p->s.size=p->q->z.text;
```

2. Разработайте макрос **ltoa3(x)** который возвращает символ, расположенный во втором слева байте длинного целого **x**.
3. Разработайте функцию, которая разместит все аргументы командной строки в дисковом файле, по одному аргументу в строке. Последний аргумент есть имя самого дискового файла.

Вариант 12

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
struct { int a:8; int b:4; } r[256][512];
```

2. Разработайте макрос **type(x,y)** который объявляет **x** как некий тип с именем **y**.
3. Разработайте функцию, которая распечатает в восьмиричном формате каждый из байтов своего аргумента типа длинное целое.

Вариант 13

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
enum z { a, b, c=20, d, e } *p;
```

2. Разработайте макрос **newtype(x,y)** который вводит новый тип **y,** являющийся указателем на известный тип **x**.
3. Разработайте функцию, которая вставляет символ в строку в указанной позиции, раздвигая строку. Используйте динамическую память.

Вариант 14

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
typedef void *paint(char,void*);
```

2. Разработайте макрос **ispunct(c)** который проверяет, является ли символ **c** знаком пунктуации.
3. Разработайте функцию, которая сравнивает содержимое двух дисковых файлов и выдает сообщение, если есть хотя бы одно несовпадение. Имена файлов заданы в командной строке.

Вариант 15

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
double*(*(*p)())();
```

2. Разработайте макрос **max3(a,b,c)** который выбирает наибольший из трех своих параметров.
3. Разработайте функцию, которая размещает аргументы командной строки в динамической памяти в виде списка.

Вариант 16

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
typedef char*z(char*);
```

2. Разработайте макрос **buffer(n)** который резервирует в памяти буфер емкостью **n** байтов и при каждом обращении выдает адрес очередного байта или **NULL**, если буфер исчерпан.
3. Разработайте функцию, которая вычисляет длину дискового файла в байтах. Имя файла задано в командной строке.

Вариант 17

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
void*(*(*(*y)[])(long*))();
```

2. Разработайте макрос **openfile(name)** который открывает файл с именем **name**.
3. Разработайте функцию, которая сравнивает две строки, пропуская при этом все встречающиеся пробелы и табуляции.

Вариант 18

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
char*(*(*(*z)(double*))[])[];
```

2. Разработайте макрос **tolower(c)** который преобразует символ **c** в нижний регистр.
3. Разработайте функцию, которая удалит из текстового файла все директивы препроцессора. Имя файла задано в командной строке.

Вариант 19

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
void*(*(*(*v)[])(void))(char*);
```

2. Разработайте макрос **isalnumrus(c)** который проверяет, принадлежит ли символ **c** множеству цифр и русских букв.
3. Разработайте функцию, которая выводит на печать каждый тринадцатый символ дискового файла. Имя файла задано в командной строке.

Вариант 20

1. Интерпретируйте следующее выражение в соответствии с правилами языка **Си**:

```
long*(*(*w)[])(void);
```

2. Разработайте макрос **toupper(c)** который преобразует символ **c** в верхний регистр.

Разработайте функцию, которая находит последнее

Контрольная работа №2

Вариант 1

1. Разработайте класс объектов типа **«текстовые файлы на жестком диске, которые способны отображать своё содержимое на экране».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **==, =, !=.** Операцию == определите полностью для случая, когда объекты являются строками символов.

Вариант 2

1. Разработайте класс объектов типа **«текстовые файлы на жестком диске, которые способны распечатывать своё содержимое на принтере»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **==, !=, ~.** Операцию ~ определите полностью для случая, когда объекты имеют тип **unsigned int.**

Вариант 3

1. Разработайте класс объектов типа **«строки, которые способны отображать себя в заданном месте экрана заданным шрифтом и цветом»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **&&, ||, ~.** Операцию **~** определите полностью для случая, когда объекты имеют тип **signed char.**

Вариант 4

1. Разработайте класс объектов типа **«папки (директории) файловой системы Windows, которые способны отображать своё содержимое на экране»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **+, -, !=.** Операцию **!=** определите полностью для случая, когда объекты являются строками символов.

Вариант 5

1. Разработайте класс объектов типа **«прямоугольные графические образы (фотографии), которые способны отображать себя на экране и сохранять на жестком диске».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **+=, -=, !.** Операцию **!** определите полностью для случая, когда объекты имеют тип **double.**

Вариант 6

1. Разработайте класс объектов типа **«графические образы произвольной формы, которые способны отображать себя на экране, перемещаться по экрану, масштабироваться и поворачиваться на 90 град.»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции присваивания и потокового ввода/вывода. Операцию присваивания определите полностью для случая, когда объекты являются строками символов.

Вариант 7

1. Разработайте класс объектов типа **«массивы, которые способны организовываться в линейный список и обеспечивающие итератор для этого списка».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции +, ==, !=. Операцию == определите полностью для случая, когда объекты являются файлами.

Вариант 8

1. Разработайте класс объектов типа **«строки, которые способны обнаружить все свои появления в текстовом файле на жестком диске».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **++, ==, --.** Операцию **--** определите полностью для случая, когда Ваши объекты являются массивами объектов типа **int.**

Вариант 9

1. Разработайте класс объектов типа **«эллипсы, которые способны отображать себя на экране внутри прямоугольных областей заданного размера»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **%, *=, /=.** Операцию **/=** определите полностью для случая, когда объекты имеют тип **float.**

Вариант 10

1. Разработайте класс объектов типа **«окружности, которые способны перемещаться по экрану и отражаться от границ экрана (игра «рикошет»)».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **++, --, =.** Операцию ++ определите полностью для случая, когда объекты являются элементами линейного списка.

Вариант 11

1. Разработайте класс объектов типа **«одномерные массивы с итераторами, которые способны контролировать правильность задания своих индексов».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **>>=, <<=, ~.** Операцию >>= определите полностью для случая, когда объекты имеют тип **char.**

Вариант 12

1. Разработайте класс объектов типа **«прямоугольные области, содержащие внутри себя текст, которые способны клонировать себя (размножаться) на экране».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **=, +, ->.** Операцию -> определите полностью для случая, когда объекты имеют тип **struct.**

Вариант 13

1. Разработайте класс объектов типа **«отрезки прямой на экране, которые способны перемещаться по экрану и вращаться относительно своего центра».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **==, [], !=.** Операцию **[]** определите полностью для случая, когда Ваши объекты являются массивами объектов типа **float.**

Вариант 14

1. Разработайте класс объектов типа **«точки на экране, которые способны перемещаться по экрану и выдавать свои координаты в декартовой и полярной системе координат»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **&=,** ^= и операции потокового ввода/вывода**.** Операции потокового ввода/вывода определите полностью для случая, когда объекты являются строками символов.

Вариант 15

1. Разработайте класс объектов типа **«символы, которые способны подсчитать количество своих появлений в текстовом файле на жестком диске»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции ***, /, %.** Операцию **%** определите полностью для случая, когда объекты имеют тип **int.**

Вариант 16

1. Разработайте класс объектов типа **«файлы, которые способны самостоятельно перемещаться по папкам (директориям) файловой системы Windows».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **+, -, *.** Операцию ***** определите полностью для случая, когда Ваши объекты являются массивами объектов типа **double.**

Вариант 17

1. Разработайте класс объектов типа **«файлы, которые способны самостоятельно распознавать и обрабатывать ошибки ввода/вывода при операциях над собой»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **>>, <<, +.** Операцию **+** определите полностью для случая, когда Ваши объекты являются файлами на жестком диске.

Вариант 18

1. Разработайте класс объектов типа **«файлы, которые способны самостоятельно дописывать своё содержимое в конец других файлов на жестком диске».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **++, --, !=.** Операцию **++** определите полностью для случая, когда Ваши объекты являются элементами линейного списка.

Вариант 19

1. Разработайте класс объектов типа **«файлы, которые способны клонировать себя (размножаться) в файловой системе Windows, меняя при этом своё имя».** При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **+, -, *, %.** Операцию **%** определите полностью для случая, когда объекты имеют тип **signed char.**

Вариант 20

1. Разработайте класс объектов типа **«объекты, которые способны клонировать себя (размножаться) в динамической памяти»**. При необходимости приведите схему наследования. Снабдите Ваше решение комментариями.
2. Опишите абстрактный класс объектов, для которых определены операции **!, ~, &.** Операцию **&** определите полностью для случая, когда объекты имеют тип **float.**

Критерии оценки:

Оценка,%	Характеристики ответа студента
76 – 100%	– знания отличаются глубиной и содержательностью, даны логично построенные, полные, исчерпывающие ответы на вопросы; – обучающийся демонстрирует способность к анализу практических ситуаций, оперирует профессиональными терминами; – обучающийся умеет высказывать и обосновать свои суждения;
51 – 75%	– знания имеют достаточный содержательный уровень, однако имеются определенные затруднения в ответе на уточняющие вопросы; – несущественные ошибки в определении понятий и категорий и т. п., кардинально не меняющие суть изложения;
26 – 50%	– знания имеют фрагментарный характер, имеются определенные неточности и погрешности в формулировках, возникают затруднения при ответе на уточняющие вопросы; – при ответе на вопросы обучающийся не может обосновать закономерности, принципы, объяснить суть явления; допущены фактические ошибки; – обучающийся продемонстрировал слабое умение формулировать выводы и обобщения,
0 – 25%	– обучающийся слабо ориентируется в теме задания, затрудняется дать ответ или дает неверные ответы на поставленные вопросы; – допущены существенные фактические ошибки при ответах на вопросы.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский ядерный университет «МИФИ»

Обнинский институт атомной энергетики –

филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»

(ИАТЭ НИЯУ МИФИ)

Отделение Интеллектуальных кибернетических систем

Темы групповых и/или индивидуальных творческих заданий/проектов

по дисциплине «Объектно-ориентированное программирование»

Лабораторная работа №1. Двоичное дерево

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cu*, которое реализует динамическую структуру данных (контейнер) типа «двоичное дерево». Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление, извлечение и поиск элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- вычисление количества элементов в контейнере;
- «очистка» контейнера, т.е. удаление всех элементов;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cu*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;

- область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.

3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Си*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Дек

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Си*, которое реализует динамическую структуру данных (контейнер) типа «Дек» (он же - двусторонний стек). Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- поиск, добавление и извлечение элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- удаление из контейнера дублирующих элементов;
- вычисление количества элементов в контейнере;
- реверс контейнера (первый элемент контейнера становится последним, второй элемент становится предпоследним и т.д.);
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Си*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.

6. Листинг разработанного авторского кода на языке *Си*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Динамический массив (вектор)

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Си*, которое реализует динамическую структуру данных (контейнер) типа «Динамический массив (вектор)». Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- поиск, добавление и извлечение элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- удаление из контейнера дублирующих элементов;
- вычисление количества элементов в контейнере;
- сортировка элементов контейнера;
- объединение, пересечение и вычитание контейнеров;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Си*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Си*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Карта

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Карта». Здесь под «картой» понимается ассоциативный массив пар элементов, состоящих из ключей и соответствующих им значений. Ключи должны быть уникальны. Порядок следования элементов определяется ключами. Элементами контейнера являются такие пары: целый ключ + значение в виде строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление и извлечение элементов контейнера;
- обход всех элементов контейнера (итератор);
- вычисление количества элементов в контейнере;
- объединение и пересечение контейнеров;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Красно-черное дерево

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Красно-черное дерево». Красно-чёрное дерево обладает следующими свойствами: а) корень должен быть чёрным; б) потомки красного узла должны быть чёрными; в) на всех ветвях дерева, ведущих от его корня к листьям, число чёрных узлов одинаково. Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление, извлечение и поиск элементов контейнера;
- вычисление количества элементов в контейнере;
- «очистка» дерева, т.е. удаление из контейнера всех элементов;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Линейный двусвязный список

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Линейный двусвязный список». Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- поиск, добавление и извлечение элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- удаление из контейнера дублирующих элементов;
- вычисление количества элементов в контейнере;
- сортировка элементов контейнера;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Множество

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Множество». Здесь под «множеством» понимается упорядоченное множество уникальных элементов. Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление, извлечение и поиск элементов контейнера;
- обход всех элементов контейнера (итератор);
- вычисление количества элементов в контейнере;
- объединение, пересечение и вычитание множеств;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Очередь с приоритетами

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Очередь с приоритетами». Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- поиск, добавление и извлечение элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- удаление из контейнера дублирующих элементов;
- вычисление количества элементов в контейнере;
- «очистка» контейнера, т.е. удаление всех элементов;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Хэш-таблица с открытой адресацией

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Cи*, которое реализует динамическую структуру данных (контейнер) типа «Хэш-таблица с открытой адресацией». Хэш-таблица с открытой адресацией состоит из массива и некоторого алгоритма разрешения коллизий. Элементами контейнера являются строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление, извлечение и поиск элементов контейнера;
- обход всех элементов контейнера (итератор);
- вычисление количества элементов в контейнере;
- «очистка» контейнера, т.е. удаление всех его элементов;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Cи*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Cи*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Циклический двусвязный список

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Си*, которое реализует динамическую структуру данных (контейнер) типа «Циклический двусвязный список». Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- поиск, добавление и извлечение элементов контейнера;
- обход всех элементов контейнера в прямом и обратном направлениях (итератор);
- удаление из контейнера дублирующих элементов;
- вычисление количества элементов в контейнере;
- сортировка элементов контейнера;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Си*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Си*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №1. Частотный словарь

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *Си*, которое реализует динамическую структуру данных (контейнер) типа «Частотный словарь». Частотный словарь содержит уникальные «слова» и информацию о том, сколько раз данное слово помещалось в контейнер. Здесь под «словами» понимаются строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

- создание и уничтожение контейнера;
- добавление и поиск элементов контейнера;
- обход всех элементов контейнера (итератор);
- вычисление количества элементов в контейнере;
- «очистка» контейнера, т.е. удаление всех его элементов;
- сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Ограничения. Реализуйте простейший проект типа «приложение командной строки» (т.е. без оконного интерфейса). Средства *C++* (объекты, классы, шаблоны классов) использовать не следует. Готовые контейнерные классы из библиотеки *STL* также использовать не следует. Разработайте контейнер самостоятельно на языке *Си*.

Рекомендации. Начните работу с изучения пункта меню «*Справка*» в *MS Visual Studio*. Создайте самостоятельно простейшее приложение типа «*Hello, world*». Затем найдите и изучите в рекомендованной литературе и в документации *MS Visual Studio* описания и примеры реализаций данной структуры данных. Обдумайте и обсудите с преподавателем алгоритмы, состав функций, интерфейс и общую структуру программы. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
   - рисунки, на которых изображена структура данных и поясняются основные операции над ней;
   - описание алгоритмов, которые используются при работе с контейнером;
   - область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
3. Структурную схему программы, на которой показаны все реализованные функции.
4. Руководство пользователя (как пользоваться программой, возможные ошибки).
5. Структуру папок разработанного проектного решения.
6. Листинг разработанного авторского кода на языке *Си*. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №2. *Windows Forms + STL*

Постановка задачи. Разработайте в *MS Visual Studio* программное решение на языке *C++* или *C#* (на ваш выбор), которое реализует для контейнера, разработанного Вами в Лабораторной работе №1, оконный пользовательский интерфейс. Используйте методы визуального программирования и классы библиотеки *Windows Forms*. Собственно контейнер реализуйте на основе шаблонов классов библиотеки *STL*. Контейнер сделайте полиморфным, чтобы он работал как минимум со следующими объектами: строки, целые числа, структуры.

Для решения задачи:

1. Изучите библиотеку базовых классов *Windows Forms.*
2. Изучите примеры использования классов *Windows Forms* для реализации оконного интерфейса. Разберитесь, как создавать обработчики событий.
3. Разработайте эскизы окон, которые Вам понадобятся, и схему управления ими.
4. Изучите библиотеку шаблонов классов *STL (Standard Template Library).*
5. Выберите в *STL* подходящие шаблоны классов, обоснуйте свой выбор.
6. Обсудите и согласуйте с преподавателем результаты пунктов 3 и 5.
7. Реализуйте на практике соответствующее программное решение.

Рекомендации. Начните работу с пробных проектов, чтобы поэкспериментировать с визуальными элементами *Windows Forms* и с «готовыми» контейнерами *STL*. Создайте несколько простых обработчиков событий. Убедитесь, что Вы понимаете структуру и логику работы приложения. Это важно, поскольку каркас приложения (т.е. существенную часть кода) *MS Visual Studio* создает автоматически. Обдумайте и обсудите с преподавателем пользовательский интерфейс Вашего приложения, а также способы обеспечения полиморфизма контейнера. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание возможностей библиотек классов *Windows Forms* и *STL*, в том числе:
   - рисунки, на которых изображены схемы наследования классов этих библиотек;
   - описание тех классов, которые были использованы в работе (данные, методы);
   - обоснование выбора конкретных классов для реализации приложения.
3. Эскизы задействованных окон, визуальных элементов и схему управления ими.
4. Описание механизма обеспечения полиморфизма контейнера.
5. Иерархию классов разработанного приложения.
6. Листинг разработанного оригинального авторского кода. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность программного решения и ответить на вопросы преподавателя.

Лабораторная работа №3. *Visual Paradigm + C#*

(выполняется проектной группой)

Постановка задачи. Спроектируйте в *Visual Paradigm* и реализуйте в *MS Visual Studio* объектно-ориентированное приложение на языке *C#*, которое представляет собой упрощенную, но действующую программную модель некоторой системы или процесса реального мира. Приложение снабдите надлежащим пользовательским интерфейсом с тем, чтобы можно было задавать исходные данные для моделирования, наблюдать за ходом процесса и просматривать итоговые результаты.

Список предметных областей для моделирования:

Лифт; Автосервис; Театральная касса; Аптека; Книжный магазин; Деканат;

Касса универсама; Регистратура поликлиники; Магазин компьютерной техники;

Бензозаправочная станция; Центр мобильной связи; Компьютерный класс;

Библиотека; Маршрутное такси; Интернет-магазин; Туристическое агентство.

Список является открытым, можно предлагать собственные объекты, системы, процессы.

Для решения задачи:

1. Изучите нотации *UML (Unified Modeling Language)*, которые применяются для построения диаграмм прецедентов и диаграмм классов.
2. Осуществите анализ и объектную декомпозицию выбранной предметной области (системы, процесса), применив разумные упрощения.
3. Спроектируйте в *Visual Paradigm* диаграммы прецедентов и диаграммы классов, обсудите их в проектной группе, а затем с преподавателем.
4. Сгенерируйте в *Visual Paradigm* необходимые классы и интерфейсы, перенесите их в *MS Visual Studio.*
5. Доработайте в *C#* и отладьте полученный код. Для организации коллективной работы используйте сервер проектов *TFS*. Убедитесь, что созданное приложение моделирует предметную область правдоподобным образом.

Рекомендации. Начните работу с пробных проектов, чтобы поэкспериментировать с *Visual Paradigm* и языком *C#*. Обратите внимание, что многие предметные области могут быть описаны моделями, которые изучаются в курсе «Моделирование систем». Активно применяйте контейнеры для работы с данными. Для моделирования случайных событий используйте генераторы случайных чисел. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание предметной области, использованных упрощений и ограничений.
3. Диаграммы прецедентов и диаграммы классов (отчет *Visual Paradigm*).
4. Иерархию классов для разработанного приложения.
5. Исходные данные и результаты моделирования предметной области.
6. Листинг разработанного авторского кода. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет (один от всей проектной группы), продемонстрировать на практике работоспособность программного решения и каждому из студентов ответить на вопросы преподавателя.

Лабораторная работа №4. *Java*

(выполняется индивидуально или проектной группой)

Постановка задачи. Реализуйте в *NetBeans IDE* на языке *Java* апплет, который бы являлся аналогом того программного решения, которое ранее было разработано Вами в Лабораторной работе 2 или 3 (на выбор). Апплет - это *web*-приложение, которое запускается с сервера, но выполняется на клиентской машине в окне браузера. Чтобы создать пользовательский интерфейс апплета, используйте библиотеку визуальных компонентов *Swing*.

Для решения задачи:

1. Создайте в *NetBeans IDE* проект, пригодный для разработки апплетов. Чтобы узнать, как это сделать, изучите «*Help*» и соответствующие примеры в *NetBeans*.
2. Разработайте в *NetBeans IDE* код апплета. Для создания оконного интерфейса используйте приемы визуального программирования, как в *MS Visual Studio.*
3. Изучите и при необходимости отредактируйте *html*-файл апплета (он создается автоматически).
4. Отладьте апплет в автономном режиме (на своем рабочем компьютере).
5. Перенесите необходимые файлы разработанного апплета на тот компьютер, где установлен *web*-сервер *GlassFish*, и разместите и в нужном домене. Запустите сервер *GlassFish.*
6. Вернитесь за свой компьютер и запустите на нем браузер, указав в адресной строке *IP* того компьютера, на котором работает сервер *GlassFish*, а также номер рабочего порта сервера и имя Вашего апплета. Убедитесь в том, что апплет загружается по сети и корректно работает в окне браузера на Вашей машине.

Рекомендации. Начните работу с простых апплетов, чтобы поэкспериментировать с *NetBeans IDE*, языком *Java* и сервером *GlassFish*. Обратите внимание на то, что апплеты не могут выполнять многие действия, которые доступны обычным приложениям. В частности, для апплетов запрещены операции с жестким диском того компьютера, на котором они выполняются. Возникающие затруднения пытайтесь преодолеть самостоятельно, потом обращайтесь за помощью.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Структуру папок проекта.
3. Иерархию классов для разработанного апплета.
4. Схему взаимодействия клиентской и серверной машины при работе апплета.
5. Листинг разработанного авторского кода. Код должен быть надлежащим образом структурирован и снабжен комментариями.

Для успешной сдачи лабораторной работы необходимо представить письменный отчет, продемонстрировать на практике работоспособность апплета и ответить на вопросы преподавателя.

Критерии оценки:

Оценка,%	Характеристики ответа студента
76 – 100%	– знания отличаются глубиной и содержательностью, даны логично построенные, полные, исчерпывающие ответы на вопросы; – обучающийся демонстрирует способность к анализу практических ситуаций, оперирует профессиональными терминами; – обучающийся умеет высказывать и обосновать свои суждения;
51 – 75%	– знания имеют достаточный содержательный уровень, однако имеются определенные затруднения в ответе на уточняющие вопросы; – несущественные ошибки в определении понятий и категорий и т. п., кардинально не меняющие суть изложения;
26 – 50%	– знания имеют фрагментарный характер, имеются определенные неточности и погрешности в формулировках, возникают затруднения при ответе на уточняющие вопросы; – при ответе на вопросы обучающийся не может обосновать закономерности, принципы, объяснить суть явления; допущены фактические ошибки; – обучающийся продемонстрировал слабое умение формулировать выводы и обобщения,
0 – 25%	– обучающийся слабо ориентируется в теме задания, затрудняется дать ответ или дает неверные ответы на поставленные вопросы; – допущены существенные фактические ошибки при ответах на вопросы.