Приложение 1

МИНОБРНАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт прикладной математики и компьютерных наук

УТВЕРЖДАЮ
Директор института прикладной
математики и компьютерных наук
_____А.В. Замятин
«_11_» _ноября_____ 2021 г.

Фонд оценочных средств по дисциплине

Объектно-ориентированное программирование

Направление подготовки
01.03.02 Прикладная математика и информатика
Профиль: **Прикладная математика и информатика**

Томск–2021

ОС составила:
канд. физ.-мат. наук, доцент
доцент кафедры компьютерной безопасности _ Е.Г. Пахомова

Рецензент:
заведующий кафедрой компьютерной безопасности,
канд. техн. наук, доцент С.А. Останин

Фонд оценочных средств одобрен на заседании учебно-методической комиссии института прикладной математики и компьютерных наук (УМК ИПМКН).

Протокол от 17 июня 2021 г. № 05

Председатель УМК ИПМКН,
д-р техн. наук, профессор С.П. Сущенко

Оценочные средства (ОС) являются элементом системы оценивания сформированности компетенций у обучающихся в целом или на определенном этапе ее формирования.

ОС разрабатывается в соответствии с рабочей программой (РП) дисциплины.

1. Компетенции и результаты обучения, формируемые в результате освоения дисциплины

Компетенция	**Индикатор компетенции**	**Код и наименование результатов обучения** (планируемые результаты обучения, характеризующие этапы формирования компетенций)	**Критерии оценивания результатов обучения**			
			Отлично	Хорошо	Удовлетвори-тельно	Неудовлетвори-тельно
ОПК-2. Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач.	ИОПК-2.1. Обладает навыками объектно-ориентированного программирования для решения прикладных задач в профессиональной деятельности.	ОР-ОПК2-2.1. Обучающийся сможет: - применять объектно-ориентированное программирование для решения задач в профессиональной деятельности	Уверенно применяет объектно-ориентированное программирование для решения задач в профессиональной деятельности	Применяет объектно-ориентированное программирование для решения задач в профессиональной деятельности	Не уверенно применяет объектно-ориентированное программирование для решения задач в профессиональной деятельности	Не может применять объектно-ориентированное программирование для решения задач в профессиональной деятельности
	ИОПК-2.2. Проявляет навыки использования основных языков программирования, основных методов разработки программ, стандартов оформления программной документации.	ОР-ОПК2-2.2. Обучающийся сможет: - использовать основные методы разработки программ - оформлять программную документацию	- Уверенно использует основные методы разработки программ. - Грамотно оформляет программную документацию	- Использует основные методы разработки программ. - Умеет оформлять программную документацию	- Не уверенно использует основные методы разработки программ. - Допускает ошибки при оформлении программной документации	- Не может использовать основные методы разработки программ. - Не может грамотно оформлять программную документацию
	ИОПК-2.3. Демонстрирует умение отбора среди существующих математических	ОР-ОПК2-2.3. Обучающийся сможет: - анализировать поставленную прикладную задачу	- Проводит всесторонний анализ поставленной прикладной	- Проводит анализ поставленной прикладной задачи	- Анализ поставленной прикладной задачи является неполным;	- Не может провести анализ поставленной прикладной задачи

		- подбирать для ее решения подходящие математические методы	задачи - подбирает для ее решения оптимальные математические методы	- подбирает для ее решения достаточно оптимальные математические методы	- предложенный для ее решения метод, не является оптимальным	- Не может подобрать для ее решения подходящие математические методы
	ИОПК-2.4. Демонстрирует умение адаптировать существующие математические методы для решения конкретной прикладной задачи.	ОР-ОПК2-2.4. Обучающийся сможет: - адаптировать существующие математические методы для решения конкретной прикладной задачи	Уверенно адаптирует существующие математические методы для решения конкретной прикладной задачи	Адаптирует существующие математические методы для решения конкретной прикладной задачи	Не уверенно адаптирует существующие математические методы для решения конкретной прикладной задачи	Не может адаптировать существующие математические методы для решения конкретной прикладной задачи
ОПК-4. Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.	ИОПК-4.1. Обладает необходимыми знаниями в области информационных технологий, в том числе понимает принципы их работы.	ОР-ОПК4-4.1. Обучающийся сможет: - понимать принципы работы современных информационных технологий	В полной мере понимает принципы работы современных информационных технологий	Понимает принципы работы современных информационных технологий	Не вполне понимает принципы работы современных информационных технологий	Не понимает принципы работы современных информационных технологий
	ИОПК-4.2. Применяет знания, полученные в области информационных технологий, при решении задач профессиональной деятельности.	ОР-ОПК4-4.2. Обучающийся сможет: - применять знания, полученные в области информационных технологий, для решения задач профессиональной деятельности	Грамотно и свободно применяет знания, полученные в области информационных технологий для решения задач профессиональной деятельности	Применяет знания, полученные в области информационных технологий для решения задач профессиональной деятельности	Не уверенно применяет знания, полученные в области информационных технологий для решения задач профессиональной деятельности	Не может применять знания, полученные в области информационных технологий для решения задач профессиональной деятельности
	ИОПК-4.3. Использует современные информационные технологии на всех этапах	ОР-ОПК4-4.3. Обучающийся сможет: - применять современные информационные технологии на всех этапах профессиональной деятельности	Уверенно применяет современные информационные технологии на всех этапах профессиональной деятельности	Применяет современные информационные технологии на всех этапах профессиональной деятельности	Не уверенно применяет современные информационные технологии на некоторых этапах профессиональной деятельности	Не может применять современные информационные технологии в профессиональной деятельности

ОПК-5. Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения.	ИОПК-5.1. Обладает необходимыми знаниями алгоритмов, приницпов разработки алгоритмов и компьютерных программ.	ОР-ОПК5-5.1. Обучающийся сможет: - использовать существующие алгоритмы и компьютерные программы для решения задач профессиональной деятельности	Уверенно применяет существующие алгоритмы и компьютерные программы	Применяет существующие алгоритмы и компьютерные программы	Не уверенно применяет некоторые алгоритмы и компьютерные программы.	Не может применять существующие алгоритмы и компьютерные программы
	ИОПК-5.2. Разрабатывает алгоритмы и компьютерные программы для решения задач профессиональной деятельности.	ОР-ОПК5-5.2. Обучающийся сможет: - разрабатывать алгоритмы и компьютерные программы для решения задач профессиональной деятельности	Грамотно и уверенно разрабатывает алгоритмы и компьютерные программы для решения задач профессиональной деятельности	Способен разрабатывать алгоритмы и компьютерные программы для решения задач профессиональной деятельности	Способен разрабатывать алгоритмы и компьютерные программы для решения большинства задач профессиональной деятельности	Не способен разрабатывать алгоритмы и компьютерные программы для решения задач профессиональной деятельности
ПК-2. Способен формализовать и алгоритмизировать поставленную задачу, написать программный код, а также верифицировать работоспособность программного обеспечения и исправить дефекты.	ИПК-2.1. Осуществляет построение формальной модели и алгоритма для поставленной задачи, написание программного кода с использованием языков программирования, верификацию работоспособности программного обеспечения и исправление дефектов.	ОР-ПК2-2.1. Обучающийся сможет: - строить формальную модель для поставленной задачи; - писать программный код; - верифицировать работоспособность программного обеспечения и исправлять дефекты	- Уверенно строит формальную модель для поставленной задачи; - грамотно пишет программный код; - уверенно верифицирует работоспособ-ность программного обеспечения и исправляет дефекты	- Способен строить формальную модель для поставленной задачи; - способен писать программный код; - способен верифицировать работоспособ-ность программного обеспечения и исправлять дефекты	- Способен строить формальную модель для большинства поставленных задач; - в большинстве случаев способен писать программный код; - в большинстве случаев способен верифицировать работоспособность программного обеспечения и исправлять дефекты	- Не способен строить формальную модель для поставленной задачи; - не способен писать программный код; - не способен верифицировать работоспособ-ность программного обеспечения и исправлять дефекты

	ИПК-2.2. Осуществляет оформление программного кода в соответствии с установленными требованиями, разработку процедур верификации работоспособности и измерения характеристик программного обеспечения, разработку тестовых наборов данных.	ОР-ПК2-2.2. Обучающийся сможет: - оформлять программный код в соответствии с установленными требованиями; - разрабатывать процедуры верификации работоспособности программного обеспечения; - разрабатывать наборы тестовых данных	- Грамотно оформляет программный код в соответствии с установленными требованиями; - уверенно разрабатывает процедуры верификации работоспособности программного обеспечения; - уверенно разрабатывает наборы тестовых данных	- Способен оформлять программный код в соответствии с установленными требованиями; - способен разрабатывать процедуры верификации работоспособности программного обеспечения; - способен разрабатывать наборы тестовых данных	- В большинстве случаев способен оформлять программный код в соответствии с установленными требованиями; - в большинстве случаев способен разрабатывать процедуры верификации работоспособности программного обеспечения; - в большинстве случаев способен разрабатывать наборы тестовых данных	- не способен оформлять программный код в соответствии с установленными требованиями; - не способен разрабатывать процедуры верификации работоспособности программного обеспечения; - не способен разрабатывать наборы тестовых данных
	ИПК-2.3. Осуществляет работу с системой контроля версий, рефакторинг и оптимизацию программного кода.	ОР-ПК2-2.3. Обучающийся сможет: - осуществлять работу с системой контроля версий; - осуществлять рефакторинг и оптимизацию программного кода	- Уверенно осуществляет работу с системой контроля версий; - уверенно осуществляет рефакторинг и оптимизацию программного кода	- Способен осуществлять работу с системой контроля версий; - способен осуществлять рефакторинг и оптимизацию программного кода	- В большинстве случаев способен осуществлять работу с системой контроля версий; - в большинстве случаев способен осуществлять рефакторинг и оптимизацию программного кода	- Не способен осуществлять работу с системой контроля версий; - не способен осуществлять рефакторинг и оптимизацию программного кода

2. Этапы формирования компетенций и виды оценочных средств

№	Этапы формирования компетенций (разделы дисциплины)	Код и наименование результатов обучения	Вид оценочного средства (тесты, задания, кейсы, вопросы и др.)
1.	Тема 1. Классы и объекты	ОР-ОПК2-2.1, ОР-ОПК2-2.2, ОР-ОПК2-2.3, ОР-ОПК2-2.4, ОР-ОПК4-4.1, ОР-ОПК4-4.2, ОР-ОПК4-4.3, ОР-ОПК5-5.1, ОР-ОПК5-5.2, ОР-ПК2-2.1, ОР-ПК2-2.2, ОР-ПК2-2.3	коллоквиум
2.	Тема 2. Наследование	ОР-ОПК2-2.1, ОР-ОПК2-2.2, ОР-ОПК2-2.3, ОР-ОПК2-2.4, ОР-ОПК4-4.1, ОР-ОПК4-4.2, ОР-ОПК4-4.3, ОР-ОПК5-5.1, ОР-ОПК5-5.2 ОР-ПК2-2.1, ОР-ПК2-2.2, ОР-ПК2-2.3	коллоквиум

3. Типовые контрольные задания или иные материалы, необходимые для оценки образовательных результатов обучения

3.1. Типовые задания для проведения текущего контроля успеваемости по дисциплине.

Текущий контроль успеваемости проводится в форме коллоквиума. Предполагается проведение двух коллоквиумов. На каждом коллоквиуме студенту предлагается ответить на теоретический вопрос и выполнить практическое задание.

Вопросы к первому коллоквиуму

А) Теоретическая часть.

1. Принципы объектно-ориентированного программирования, их суть. Определение класса. Объект. Сокрытие информации. Чем член-функции отличаются от обычных. Какие существуют типы доступа и чем они отличаются друг от друга. Неявный указатель this.
2. Чем член-функции отличаются от обычных. Операция ::. Чем отличаются член-функции, определенные внутри класса, от функций, определенных вне класса.
3. Конструкторы и деструкторы. Назначение конструктора. Особенности конструктора. Деструктор. Особенности деструктора.
4. Конструктор копирования. В каком случае необходим конструктор копирования и почему. Чем конструктор копирования отличается от перегрузки операции =.
5. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузки операций + и +=. Чем отличается операция + от операции +=.
6. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузка операций [ ], ( ).
7. Перегрузка операций. Перечислите правила перегрузки операций. Особенности перегрузки операции (тип). Перегрузка операции ++.
8. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузка операции =. Когда необходимо перегружать операцию =. Чем конструктор копирования отличается от перегрузки операции =.
9. Дружественность. Что может быть другом класса. Перегрузка операций потокового ввода >> и вывода <<. Что общего и в чем разница между перегрузкой операции потокового вывода << и перегрузкой операции потокового ввода >>. Можно ли перегрузить операции потокового ввода/вывода как методы класса, ответ

аргументировать. Обязательно ли использовать ссылку при перегрузке потоковых операций.

10. Массивы объектов. Какие конструкторы можно использовать для инициализации статических объектов. Можно ли явно инициализировать массив объектов, определенных в динамической памяти. Как работает конструктор для массива объектов. Как работает деструктор для массива объектов.
11. Агрегированные классы. Каким образом можно определить конструктор агрегированного класса, если член-данное в нем – указатель на объект другого (используемого) класса? Каким образом агрегированный класс может использовать член-данные используемого класса из части private.
12. Агрегированные классы. Каким образом агрегированный класс может использовать член-данные используемого класса из части private.
13. Функции- и классы-шаблоны. Что такое порожденная функция. Использование функций-шаблонов и классов-шаблонов. Описать работу компилятора.
14. Статические член-данные класса: как объявляется статическое член-данное в классе, его отличие от остальных член-данных; как инициализируется статическое член-данное, как влияет тип доступа на инициализацию статического член-данного; можно ли изменять статическое член-данное в main; как можно получить доступ к статическому член-данному в main.
15. Статические методы класса: как объявляются статические методы в классе, их назначение; как определяются статические методы вне класса; особенности статических методов класса

Б) Практическая часть.

1. Класс Circle (Круг) .
   Член-данные: double x, y (координаты центра) и double r (радиус).
   Методы: все конструкторы (если не требуется конструктора копирования – объяснить почему); ввод и вывод; нахождение площади; проверка, попадает ли заданная точка в круг.
   Перегрузка операторов: операции + и += (умножение круга на число – центр не меняется, радиус умножается на число); сравнение (== и !=; два круга равны, если равны их радиусы); потокового ввода >> и потокового вывода <<.
2. Класс FreeVector (Свободный вектор) .
   Член-данные: double x, y, z (координаты вектора).
   Методы: все конструкторы (если не требуется конструктора копирования – объяснить почему); ввод и вывод; нахождение длины вектора.
   Перегрузка операторов: операции + , += (сумма векторов); операции – , –= (разность векторов); операции * (скалярное произведение векторов); сравнения (== и !=); потокового ввода >> и потокового вывода <<.
3. Класс Complex.
   Член-данные: double x, y (действительная и мнимая часть комплексного числа).
   Методы: все конструкторы (если не требуется конструктора копирования – объяснить почему); ввод, вывод, модуль;комплексно-сопряженное;
   Перегрузка операторов: арифметические операции (+, +=, –, – =, *, *=, /, /=); сравнение (== и !=); потокового ввода >> и потокового вывода <<.
4. Класс массив Array.
   Член-данные (2 варианта):
   А) указатель на целое, количество взятой памяти (mem), число элементов массива (n);
   Б) указатель на целое, число элементов массива (памяти берётся столько же) (n);
   Методы:
   Для А – Array(int mem_=1, int n_=0) (можно добавить параметр – способ заполнения) , для Б – Array(int n_=0) (можно добавить параметр – способ заполнения);

Array(int*, int) (создание объекта класса Array из целочисленного массива); Array(const Array&); ~Array(); функции ввода/вывода; формирование случайного массива; функция поиска (на входе – ключ, на выходе – индекс, если ключ присутствует в массиве, –1, если ключа в массиве нет); функция нахождения max/min элемента (возвращает индекс элемента); добавление ключа на позицию;
удаление позиции (изменяется существующий объект, создается новый объект).
Перегрузка операторов:
= (присвоение); [ ] (возвращает ссылку); +(int key) – добавление ключа key в конец массива; +=(int key) – добавление ключа key в конец массива; +(Array) – добавление массива в конец массива; +=(Array) – добавление массива в конец массива; –(int key) – удаление ключа key (первое вхождение); –=(int key) – удаление ключа key (первое вхождение; ==, !=(сравнение); потоковый ввод/вывод;

5. Класс String
   Член-данные: char *Str (строка), int Len (размер строки Str).
   Методы: String(char *); String(int); String(const String&); ~String().
   Перегрузка операторов: = (присвоение); [ ] (возвращает ссылку); >> (потоковый ввод); << (потоковый вывод); +, += (конкатенация); == (сравнение строк).
6. Класс-шаблон Stack на основе массива.
   Член-данные: массив, размер массива, реальное число элементов стека.
   Методы: Stack(int k=50); Stack(const Stack<T> &); ~ Stack(); проверка на пустоту; проверка на заполненность; добавить в стек элемент; удалить из стека элемент; взять значение с вершины стека;
   Перегрузка операторов: = (присвоение); потоковый вывод << .

Вопросы ко второму коллоквиуму

А) Теоретическая часть.

1. Базовый и порожденный классы. Тип доступа protected. Типы наследования public, protected и private. Ограничения наследования.
2. Простое наследование. Принцип доминирования в иерархии наследования.
3. Конструктор порожденного класса. Его вид. Деструктор порожденного класса.
4. Стандартные преобразования при наследовании.
5. Множественное наследование. Прямые и не прямые базовые классы. Виртуальный базовый класс. Особенности инициализации его ч/данных.
6. Полиморфизм. Раннее и позднее связывание.
7. Виртуальные функции. Чистые виртуальные функции и абстрактный базовый класс.
8. Правила определения виртуальных функций: описание, определение, виртуальные функции порожденного класса.
9. Правила определения виртуальных функций: виртуальные и невиртуальные вызовы.
10. Правила определения виртуальных функций: виртуальные операции, приватные функции.
11. Механизм позднего связывания.
12. Библиотека STL – БСШ. Определение контейнера. Структура БСШ.
13. Библиотека стандартных шаблонов. Шаблоны vector, list, set, stack, queue
14. Класс «Факультет» (классы Person, Student, Teacher, Skills, Faculty)

Б) Практическая часть.

1. Класс List (список). Может быть однонаправленный или двунаправленый, с фиктивной головой или нет.
   2 независимых класса – узел (Node) и список (могут быть друзьями).
   Член-данные для списка: указатель на голову, указатель на хвост (для двунаправленного)
   Методы: List (); List (int); List (int*, int) (из массива делаем список); List (const List &); ~List(); Node* FindKey(int) (на входе – ключ, на выходе – указатель на узел, если ключ

в списке нашли, NULL – если ключа в списке нет); функции ввода/вывода (можно только перегрузить потоковые); сортировка; добавление элемента (в голову, хвост, на позицию, после ключа), удаление элемента (из головы, хвоста, с позиции, по ключу), нахождение max/min; проверка пустоты списка; очистка списка.
Перегрузка операторов: = (присвоение); [ ] ; +(List) (в голову, в хвост); ==, != (сравнение списков); потокового ввода/вывода.

2. Классы Student и Teacher, наследники класса Person.
а) Определить родительский класс Person (член-данные: string fio, address; int year; методы: конструкторы; void Input() (ввод данных с клавиатуры); void Show() (вывод данных на консоль).
б) Член-данные класса Student: int group, kurs; string spec.
Методы класса Student: конструкторы; void Input() (ввод данных с клавиатуры); void Show() (вывод данных на консоль), Get_group(); Get_kurs();
Перегрузить потоковый вывод.
в) Член-данные класса Teacher: string kafedra, prof.
Методы класса Student: конструкторы; void Input() (ввод данных с клавиатуры); void Show() (вывод данных на консоль), Get_Kafedra(); Get_Prof();
Перегрузить потоковый вывод.
3. Агрегированный класс Faculty.
Член-данные класса Faculty: vector<Student> s; vector <Teacher> t; string long_name, short_name;
Методы: конструкторы; добавление студента/преподавателя; удаление студента/преподавателя; вывод (всех, всех студентов, всех преподавателей, конкретной группы, конкретного курса, конкретной кафедры); получить все данные студента/преподавателя; вывод общего количества студентов/преподавателей.
4. БСШ, класс vector.
Задать массив случайных целых чисел и
а) выбрать из него в другой массив уникальные числа (т.е. встречающиеся только один раз); б) выбрать из него в другой массив числа, кратные 5.
Задать массив строк и удалить из него самую длинную строку.
Задать массив чисел Фибоначчи и выбрать из него в другой массив четные числа.
5. БСШ, класс list.
Задать список фамилий и
а) Добавить перед самой длинной фамилией фамилию fam; б) отсортировать его по возрастанию; в) найти и удалить самую длинную фамилию.
Задать 2 упорядоченных списка фамилий, слить их в один общий упорядоченный список.
6. БСШ – класс set
Задать 2 предложения. Выяснить, в каком предложении разных букв больше.

3.2. Типовые задания для проведения промежуточной аттестации по дисциплине

Билет для зачета формируется из вопросов к коллоквиумам. В каждый билет включается два теоретических и два практических задания.

БИЛЕТ 1

1. Определение класса и объекта. Какие существуют типы доступа и чем они отличаются друг от друга. Отличие член-функций класса от обычных функций. Чем отличаются функции, определенные внутри класса, от функций, определенных вне класса.

2. Библиотека STL – БСШ. Определение контейнера. Структура БСШ.

3. Класс-шаблон **Stack** (стек на основе массива). Реализовать **методы**: а) конструкторы (все); б) деструктор; в) проверка на пустоту; г) добавить в стек элемент; д) перегрузить оператор потокового вывода

4. Класс List. Реализовать **методы**: а) конструкторы (все); б) деструктор; в) очистка списка; г) нахождение max; д) перегрузка потокового вывода.

БИЛЕТ 2

1. Конструкторы: назначение конструктора; особенности конструктора; количество конструкторов в классе; когда в классе необходим конструктор копирования и почему.

Деструктор: назначение деструктора, особенности деструктора, когда в классе необходим деструктор.

2. Механизм позднего связывания.

3. Класс **String** (строка). Реализовать **методы**: а) конструкторы (все); б) деструктор. Перегрузить **операторы**: в) [], г) == (сравнение строк)

4. БСШ – класс list: пример использования – заданы 2 упорядоченных списка фамилий, слить их в один общий упорядоченный список.

БИЛЕТ 3

1. Конструктор копирования. В каком случае необходим конструктор копирования и почему. Чем конструктор копирования отличается от перегрузки операции =. В каких ситуациях работает конструктор копирования.

2. Правила определения виртуальных функций: виртуальные операции, приватные функции.

3. Класс **Array** (массив). Реализовать **методы**: а) конструкторы (все); б) деструктор. Перегрузить **операторы**: в) [], г) +(Array), д) +=(Array).

4. Классы Student и Teacher. Агрегированный класс Faculty: определить функцию ««вывод преподавателей факультета по должности».

БИЛЕТ 4

1. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузки операций + и +=. Чем отличается операция + от операции +=.

2. Правила определения виртуальных функций: виртуальные и невиртуальные вызовы.

3. Класс **Array** (массив). Реализовать **методы**: а) функцию FindKey(int) (поиск ключа, на входе – ключ, на выходе – индекс (если нашли), –1 (если нет)); б) функцию FindMin() (возвращает индекс минимального элемента). Перегрузить **операторы**: в) =; г) –(int key) (удаляет первое вхождение ключа key); д) –=(int key) (удаляет первое вхождение ключа key).

4. Класс List. Реализовать **методы**: а) конструкторы (все); б) деструктор; в) удаление элемента с хвоста; г) сортировка списка; д) перегрузка оператора [].

БИЛЕТ 5

1. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузка операций ( ). Перегрузка операции ++

2. Правила определения виртуальных функций: описание, определение, виртуальные функции порожденного класса.

3. Класс **String** (строка). Реализовать **методы**: а) конструкторы (все); б) деструктор. Перегрузить **операторы**: г) =; д) потоковый ввод.

4. БСШ – класс vector: пример использования – задать массив чисел Фибоначчи, выбрать из него в другой массив четные числа.

БИЛЕТ 6

1. Перегрузка операций. Перечислите правила перегрузки операций. Когда возможно преобразование переменной стандартного типа в переменную пользовательского типа и наоборот. Особенности перегрузки операции (тип). Перегрузка операции [ ].

2. Виртуальные функции. Чистые виртуальные функции и абстрактный базовый класс.

3. Класс **Array** (массив). Реализовать **методы**: а) функцию DelPosEq(int pos) (удаление элемента с позиции pos, изменяется *this). Перегрузить **операторы**: б) +(int

key), (добавление ключа key в конец массива); в) = (присвоение); г) == (сравнение); д) потоковый ввод.

4. Классы Student и Teacher. Агрегированный класс Faculty: определить функцию «вывод всего состава факультета (студентов и преподавателей)».

БИЛЕТ 7

1. Перегрузка операций. Перечислите правила перегрузки операций. Перегрузка операции =. Когда необходимо перегружать операцию =. Чем конструктор копирования отличается от перегрузки операции =.

2. Полиморфизм. Раннее и позднее связывание.

3. Класс-шаблон **Stack** (стек на основе массива). Реализовать **методы**: а) конструкторы (все); б) деструктор; в) проверка на заполненность; г) удалить элемент из стека; д) перегрузить оператор = .

4. Класс List. Реализовать **методы**: а) конструкторы (все); б) деструктор; в) удаление элемента по указателю; г) добавление элемента в голову списка; д) перегрузка оператора =.

БИЛЕТ 8

1. Дружественность. Что может быть другом класса. Перегрузка операций потокового ввода >> и вывода <<, что общего между ними и в чем различие? Можно ли перегрузить операции потокового ввода/вывода как методы класса, ответ аргументировать. Обязательно ли использовать ссылку для второго операнда при перегрузке потоковых операций.

2. Множественное наследование. Прямые и не прямые базовые классы. Виртуальный базовый класс. Особенности инициализации его ч/данных.

3. Класс **Circle** (круг). **Член-данные:** double x, y (координаты центра) и double r (радиус). Реализовать **методы**: а) конструкторы (все), деструктор (если требуется). Если не требуется конструктора копирования – объяснить почему. Если не требуется деструктор – объяснить почему; б) нахождение площади. Перегрузить **операторы**: в) * и *= (умножение круга на число – центр не меняется, радиус умножается на число); г) потокового вывода <<.

4. БСШ – класс list: пример использования – задан список фамилий, найти и удалить самую длинную фамилию.

БИЛЕТ 9

1. Массивы объектов. Какие конструкторы можно использовать для инициализации статических массивов объектов. Можно ли явно инициализировать массив объектов, определенных в динамической области. Как работает конструктор для массива объектов (статического и динамического). Как работает деструктор для массива объектов (статического и динамического).

2. Стандартные преобразования при наследовании.

3. Класс **FreeVector** (свободный вектор). **Член-данные:** double x, y, z (координаты вектора). Реализовать **методы**: а) конструкторы (все), деструктор (если требуется). Если не требуется конструктора копирования – объяснить почему. Если не требуется деструктор – объяснить почему. б) нахождение длины вектора. Перегрузить **операторы**: в) + и += (сумма векторов); г) потокового вывода << .

4. Классы Student и Teacher. Агрегированный класс Faculty: определить функцию «вывод студентов группы k».

БИЛЕТ 10

1. Определение агрегированного класса. Как работает конструктор в агрегированном классе. Каким образом можно определить конструктор агрегированного класса, если член-данное в нем – указатель на объект другого (используемого) класса? Каким образом агрегированный класс может использовать член-данные используемого класса из части private.

2. Конструктор порожденного класса. Его вид. Деструктор порожденного класса.

3. Класс **Complex** (комплексное число). **Член-данные:** double x, y (действительная и мнимая часть числа). Реализовать **методы**: а) конструкторы (все), деструктор (если требуется). Если не требуется конструктора копирования – объяснить почему. Если не требуется деструктор – объяснить почему. б) нахождение модуля комплексного числа. Перегрузить **операторы**: в) * (произведение комплексных чисел); г) /= (частное комплексных чисел); д) потокового вывода <<.

4. Класс List. Реализовать **методы**: а) конструкторы (все); б) деструктор; в) удаление элемента с головы; г) добавление элемента после заданного узла; д) перегрузка оператора ==.

БИЛЕТ 11

1. Функции- и классы-шаблоны. Что такое порожденная функция. Использование функций-шаблонов и классов-шаблонов. Описать работу компилятора.

2. Простое наследование. Принцип доминирования в иерархии наследования.

3. Класс **String** (строка). Реализовать **методы**: а) конструкторы (все). Перегрузить **операторы**: б) +, += (конкатенация строк); в) потоковый вывод.

4. БСШ – класс set:пример использования – заданы 2 предложения. В каком предложении разных букв больше?

БИЛЕТ 12

1. Статические член-данные класса: как объявляется статическое член-данное в классе, его отличие от остальных член-данных; как инициализируется статическое член-данное, как влияет тип доступа на инициализацию статического член-данного; можно ли изменять статическое член-данное в main; как можно получить доступ к статическому член-данному в main.

Статические методы класса: как объявляются статические методы в классе, их назначение; как определяются статические методы вне класса; особенности статических методов класса.

2. Базовый и порожденный классы. Тип доступа protected. Типы наследования public, protected и private. Ограничения наследования.

3. Класс **Array** (массив).

Реализовать **методы:**

а) функцию DelPosNew(int pos) (удаление элемента с позиции pos, создается новый объект). Перегрузить **операторы**: б) +=(int key) (добавление ключа key в конец массива); в) = (присвоение); г) == (сравнение); д) потоковый вывод.

4. Классы Student и Teacher. Агрегированный класс Faculty: определить функцию «добавить преподавателя с известными данными на кафедру N».

4. Методические материалы, определяющие процедуры оценивания образовательных результатов обучения

4.1. Методические материалы для оценки текущего контроля успеваемости по дисциплине.

Коллоквиумы проводятся в письменной форме по билетам. Билет состоит из двух частей. Первая часть представляет собой теоретический вопрос, вторая – практическое задание.

По результатам коллоквиума студенту выставляются оценки «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Оценка «отлично» выставляется, если:

а) студент дал полный и развернутый ответ на теоретический вопрос;

б) код практического задания верен, оптимален (по скорости или по объему памяти), легко читаем, при написании кода использованы эффективные алгоритмы.

Оценка «**хорошо**» выставляется, если:

а) ответ студента на теоретический вопрос в целом полный, но имеются незначительные замечания;

б) код практического задания верен, но не оптимален (по скорости и по объему памяти), при написании кода использованы трудоемкие алгоритмы.

в) код практического задания верен, оптимален (по скорости или по объему памяти), легко читаем, при написании кода использованы эффективные алгоритмы, но студент выполнил не все практическое задание (но более 80%)

Оценка «**удовлетворительно**» выставляется, если:

а) ответ студента на теоретический вопрос не полный;

б) код практического задания содержит ошибки синтаксического характера.

в) студент выполнил менее 80% практического задания (но более 50%)

Оценка «**неудовлетворительно**» выставляется, если:

а) студент не ответил на теоретический вопрос или ответ студента на теоретический вопросы не полный и содержит серьезные ошибки;

б) практическое задание не выполнено или код практического задания содержит синтаксические и алгоритмические ошибки.

в) студент выполнил менее 50% практического задания.

4.2. Методические материалы для проведения промежуточной аттестации по дисциплине.

Зачет с оценкой в третьем семестре проводится в письменной форме по билетам. Билет состоит из двух частей.

Первая часть представляет собой два теоретических вопроса. Ответы на вопросы даются в развернутой форме.

Вторая часть состоит из двух практических заданий. Ответы на вопросы второй части предполагают написание программного кода для поставленной задачи и анализ его работы. Оценивается оптимальность выбранного для решения задачи алгоритма и скорость его работы.

Результаты зачета с оценкой определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Оценка «**отлично**» выставляется, если:

а) студент дал полный и развернутый ответ на теоретические вопросы;

б) код практического задания верен, оптимален (по скорости или по объему памяти), легко читаем, при написании кода использованы эффективные алгоритмы.

Оценка «**хорошо**» выставляется, если:

а) ответ студента на теоретические вопросы в целом полный, но имеются незначительные замечания;

б) код практического задания верен, но не оптимален (по скорости и по объему памяти), при написании кода использованы трудоемкие алгоритмы.

в) код практического задания верен, оптимален (по скорости или по объему памяти), легко читаем, при написании кода использованы эффективные алгоритмы, но студент выполнил не все практическое задание (но более 80%)

Оценка «**удовлетворительно**» выставляется, если:

а) ответ студента на теоретические вопросы не полный;

б) код практического задания содержит ошибки синтаксического характера.

в) студент выполнил менее 80% практического задания (но более 50%)

Оценка «**неудовлетворительно**» выставляется, если:

а) студент не ответил на теоретические вопросы или ответ студента на теоретические вопросы не полный и содержит серьезные ошибки;

б) практическое задание не выполнено или код практического задания содержит синтаксические и алгоритмические ошибки.

в) студент выполнил менее 50% практического задания.

Если в течение семестра студент посетил не менее 75% занятий и выполнил все практические задания, то он освобождается от выполнения практической части билета.