Федеральное государственное бюджетное образовательное учреждение

высшего образования

Московский государственный институт культуры

УТВЕРЖДЕНО
Председатель УМС
Библиотечно-информационного
факультета
Боронина Н.В.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДИСЦИПЛИНЫ (МОДУЛЯ)

ПРОГРАММИРОВАНИЕ

Направление подготовки/специальности (код, наименование)
09.03.02 «Информационные системы и технологии»

Профиль подготовки/специализация
Информационные системы и цифровые технологии в культуре

Квалификация (степень) выпускника
Бакалавр

Форма обучения
Очная

(РПД адаптирована для лиц
с ограниченными возможностями
здоровья и инвалидов)

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели:
изучение основных принципов процедурного и модульного программирования; обучение правилам и подходам к разработке алгоритмов, кодированию и отладке программ на языке программирования С + +, пригодных для практического применения, изучение принципов работы программного средства Dev С + + и его использование при решении прикладных задач.

Задачи:
- изучение основных синтаксических конструкций языка программирования С + +, правил и рекомендаций построения программ на указанном языке;
- изучение возможностей среды разработки программного обеспечения Dev C++;
- привитие практических умений и навыков разрабатывать алгоритмы и программы, пригодные для практического применения; привития умений писать и отлаживать коды на языке программирования С + +, тестировать работоспособность программы на указанном языке, навыков отладки и тестирования работоспособности программ;
- привитие умений и навыков применения современных информационных технологий и программных средств, в том числе отечественного производства, при решении прикладных задач различных классов, их отладки и тестирования работоспособности.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП ВО

Дисциплина «Программирование» входит в состав Блока 1 «Дисциплины (модули)» и относится к обязательной части ОПОП по направлению подготовки 09.03.02 «Информационные системы и технологии», профиль – Информационные системы и цифровые технологии в культуре.
Дисциплина «Программирование» изучается в 3, 4 семестрах. Входные знания, умения и компетенции, необходимые для изучения данного курса, формируются в процессе изучения таких дисциплин, как: «Русский язык и культура речи», «Иностранный язык», «Основы естественно-научных и общеинженерных знаний», «Математика», «Теоретические основы информатики», «Информационная культура личности», «Основы российской государственности», «Современные информационные технологии и программное обеспечение», «Вычислительные сети и системы», «Информационная безопасность и защита информации». Взаимосвязь курса с другими дисциплинами ООП способствует планомерному формированию необходимых компетенций и углубленной подготовке студентов к решению специальных практических профессиональных задач.

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Процесс освоения дисциплины направлен на формирование компетенций в соответствии с ФГОС ВО и ОПОП ВО по данному направлению подготовки (специальности) 09.03.02 «Информационные системы и технологии»:
Перечень планируемых результатов обучения по дисциплине (модулю).

Компетенция (код и наименование)	**Индикаторы компетенций**	**Результаты обучения**
ОПК-6 Способен разрабатывать алгоритмы и	ОПК-6.1 - Разрабатывает алгоритмы и программы,	Знать: теорию информационных систем, теорию баз данных, методы процедурного моделирования, основные языки

программы, пригодные для практического применения в области информационных систем и технологий	пригодные для применения в области ИС.	программирования, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий Уметь: применять языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации процессов реальной деятельности, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ, организации и эксплуатации информационных ресурсов Владеть: навыками программирования, отладки и тестирования программно-технических комплексов
ОПК-8 Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем	ОПК-8.1 - Проектирует ИС, объясняет применение для этой цели математических моделей и методов.	Знать: основы математического моделирования, методологии и средства проектирования информационных и автоматизированных систем Уметь: применять математические модели, методы и средства проектирования информационных и автоматизированных систем Владеть: навыком проектирования информационных систем в соответствие с поставленной задачей в стандартных условиях

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ *(модуля)*

4.1 Объем дисциплины (модуля)

Объем (общая трудоемкость) дисциплины «Программирование» составляет 6 зе, 216 акад. часов, из них контактных 92 акад.ч., СРС 124 акад.ч., формы контроля зачёт и зачёт с оценкой.

4.2. Структура дисциплины для очной формы обучения.

№ п/п	Тема/Раздел дисциплины	Семестр	Виды учебной работы*, включая самостоятельную работу студентов и трудоемкость (в часах)/ с указанием занятий, проводимых в интерактивных формах					Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
			Лекции	Семинары/ практические	Консультации	ИКР	СРС	
1	*Тема 1 Структура программы на языке С++*	3	10	4			16	*Практическое задание*
2	*Тема 2.* Разветвляющиеся программы	3	6	4			10	*Практическое задание*
3	*Тема 3*. Операторы цикла	3	6	2			12	*Практическое задание*
4	*Тема 4.* Пользовательские функции	3	6	2			10	*Практическое задание*
5	*Тема 5*. Особенности обработки массивов данных	3	6	2			10	*Практическое задание*
	Зачет							
6	*Тема 6*. Указатели и циклы	4	6	8			22	*Практическое задание*
7	*Тема 7*. Типы данных, определяемые пользователем	4	6	6			22	*Практическое задание*
8	*Тема 8*. Обработка файлов	4	8	8			22	*Практическое задание*
	Зачёт с оценкой	4						*Тест*
	итого:		54	38			124	

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Текущий контроль выполнения заданий (контроль формирования компетенций) осуществляется регулярно, начиная с первой недели семестра (входящий контроль). Контроль и оценивание выполнения (например, эссе) осуществляется на 2 неделе семестра. Текущий контроль освоения отдельных разделов дисциплины осуществляется при помощи контрольных работ и тестовых заданий в завершении изучения каждого раздела. Система текущего контроля успеваемости служит не только оценке уровня компетентностной подготовки обучающегося и способствует в дальнейшем наиболее качественному и объективному оцениванию его в ходе промежуточной аттестации, но и самооценке учащегося, стимулируя его усилия.

Промежуточная аттестация по дисциплине:

Промежуточная аттестация проводится в форме зачёта/зачёта с оценкой.

Примерные вопросы на контрольные работы:

Контрольная работа №1

Вариант 1

Инструкция: Задания выполняются на компьютере в среде разработки. Результатом работы является исправно работающая программа, демонстрирующая решение поставленной задачи. Код должен быть отформатирован и содержать комментарии.

Задание 1. Линейный алгоритм и ввод/вывод (5 баллов)
Напишите программу, которая запрашивает у пользователя его имя, возраст и средний балл (рейтинг). Программа должна вывести приветствие и форматированную табличку:

****
* Имя: Иван
* Возраст: 20 лет
* Рейтинг: 4.75
****
Использовать манипуляторы `setw()` для выравнивания.

Задание 2. Условный оператор (5 баллов)
Напишите программу, которая запрашивает у пользователя три числа (длины сторон треугольника). Определите, существует ли треугольник с такими сторонами (правило: каждая сторона меньше суммы двух других). Если треугольник существует, определите его тип:
- Равносторонний
- Равнобедренный
- Разносторонний
Если треугольник не существует, выведите соответствующее сообщение.

Задание 3. Циклы и массивы (5 баллов)
Сгенерируйте массив из 20 случайных чисел в диапазоне от -10 до 10. Выведите массив на экран. Найдите и выведите:
1. Сумму всех положительных элементов.
2. Произведение всех отрицательных элементов.
3. Количество нулевых элементов.

Задание 4. Функции и указатели (5 баллов)
Напишите функцию `swapMaxMin`, которая принимает массив целых чисел и его размер. Функция должна найти максимальный и минимальный элемент в массиве и поменять их местами. Используйте передачу параметров по указателю для возврата индексов найденных элементов (или используйте ссылки).
В функции `main()` продемонстрируйте работу: выведите массив до и после обмена.

Задание 5. Структуры и файлы (10 баллов)
Создайте структуру `Student`, содержащую поля:
- ФИО (строка)
- Номер группы (целое число)
- Оценки по 3 предметам (массив из 3 целых чисел)
Задачи:
1. В программе создать массив из 3 студентов и заполнить его данными (можно "зашить" в коде или ввести с клавиатуры).
2. Сохранить этот массив в текстовый файл `students.txt` в удобочитаемом формате.
3. Прочитать данные из файла обратно в новый массив.
4. Вывести на экран список студентов, у которых средний балл выше 4.0.

Критерии оценки:
«Отлично» - 26-30 баллов
«Хорошо» - 21-25 баллов
«Удовлетворительно» - 15-20 баллов
«Неудовлетворительно» - Менее 15 баллов

Вариант 2
Инструкция: Задания выполняются на компьютере в среде разработки. Результатом работы является исправно работающая программа.

Задание 1. Линейный алгоритм и ввод/вывод (5 баллов)
Напишите программу для перевода рублей в доллары и евро. Программа запрашивает:
- Количество рублей
- Курс доллара
- Курс евро

Программа выводит результат в формате:
X рублей = Y долларов = Z евро
Все значения должны выводиться с двумя знаками после запятой.

Задание 2. Условный оператор (5 баллов)
Напишите программу "Калькулятор". Пользователь вводит два числа и арифметическую операцию (+, -, *, /). Используя оператор `switch`, выполните соответствующее действие и выведите результат. Предусмотрите обработку деления на ноль и неверного символа операции.

Задание 3. Циклы и массивы (5 баллов)
Создайте массив из 15 целых чисел, введенных пользователем с клавиатуры. Выведите массив в прямом и обратном порядке. Найдите и выведите:
1. Индекс первого отрицательного элемента.
2. Сумму элементов, стоящих на четных позициях (индексы 0, 2, 4...).

Задание 4. Функции и указатели (5 баллов)
Напишите функцию `isSorted`, которая проверяет, отсортирован ли массив по возрастанию. Функция должна принимать массив и его размер, возвращать `true` или `false`. Также напишите функцию `bubbleSort`, которая сортирует массив методом пузырька.
В `main()` создайте массив, выведите результат проверки, если не отсортирован — отсортируйте и выведите отсортированный массив.

Задание 5. Структуры и файлы (10 баллов)
Создайте структуру `Book`, содержащую поля:
- Название книги
- Автор
- Год издания
- Наличие в библиотеке (булево значение)
Задачи:
1. Создайте массив из 4 книг и заполните его данными.
2. Сохраните данные в бинарный файл `books.dat`.
3. Прочитайте данные из бинарного файла и выведите на экран только те книги, которые выданы читателям (`наличие == false`).

4. Добавьте возможность пользователю ввести название книги и найти её в массиве (линейный поиск), вывести всю информацию о книге.

Контрольная работа №2
Вариант 1 (Итоговая, повышенной сложности)
Инструкция: Задания выполняются на компьютере. Работа считается зачтенной при выполнении не менее 3 заданий.

Задание 1. Динамические массивы (5 баллов)
Напишите программу, которая:
1. Запрашивает у пользователя размер массива.
2. Создает динамический массив этого размера.
3. Заполняет его случайными числами от 0 до 100.
4. Находит и выводит второй по величине элемент массива (без использования сортировки).
5. Корректно освобождает память.

Задание 2. Многомерные массивы (5 баллов)
Создайте двумерный динамический массив (матрицу) размером N x M (N и M вводятся пользователем). Заполните матрицу случайными числами. Выведите её на экран. Транспонируйте матрицу и выведите результат.
Транспонирование — замена строк на столбцы.

Задание 3. Работа со строками (5 баллов)
Напишите функцию `countWords`, которая принимает строку (массив символов) и возвращает количество слов в ней. Слова разделяются одним или несколькими пробелами. Не использовать библиотеку `string`, только `cstring` и символьные массивы.
В `main()` запросить у пользователя предложение и вывести количество слов.

Задание 4. Рекурсия (5 баллов)
Напишите рекурсивную функцию `gcd`, которая находит наибольший общий делитель двух чисел с помощью алгоритма Евклида. Продемонстрируйте её работу для двух чисел, введенных пользователем.

Задание 5. Комплексная задача: "База данных студентов" (10 баллов)
Создайте структуру `Student` с полями: ФИО, группа, 4 оценки за экзамены.
Реализуйте меню:
1. Добавить студента (данные вводятся с клавиатуры, массив структур динамический).
2. Показать всех студентов (в виде таблицы).
3. Сохранить данные в текстовый файл.
4. Загрузить данные из текстового файла.
5. Показать студентов-отличников (все оценки 5).
6. Выход.
Программа должна работать в цикле до выбора пункта "Выход". Данные хранятся в динамическом массиве, который расширяется при добавлении новых студентов.

Вариант 2 (Итоговая, повышенной сложности)
Инструкция: Задания выполняются на компьютере. Работа считается зачтенной при выполнении не менее 3 заданий.

Задание 1. Динамические массивы (5 баллов)
Напишите программу, которая:

1. Создает динамический массив на 10 элементов (заполнить случайными).
2. Выводит исходный массив.
3. Запрашивает у пользователя индекс элемента, который нужно удалить.
4. Создает НОВЫЙ динамический массив размером на 1 меньше и копирует туда все элементы, кроме удаляемого.
5. Выводит получившийся массив.
6. Освобождает память.

Задание 2. Многомерные массивы (5 баллов)
Создайте двумерный динамический массив (матрицу) размером N x N (квадратную). Заполните её случайными числами. Выведите матрицу. Найдите сумму элементов на главной диагонали и произведение элементов на побочной диагонали.

Задание 3. Работа со строками (5 баллов)
Напишите функцию `isPalindrome`, которая проверяет, является ли строка палиндромом (читается одинаково слева направо и справа налево). Функция должна игнорировать пробелы и регистр букв. Использовать символьные массивы.
Пример: "А роза упала на лапу Азора" — палиндром.

Задание 4. Рекурсия (5 баллов)
Напишите рекурсивную функцию `power`, которая возводит число в степень. Функция принимает основание (double) и показатель степени (целое неотрицательное число). Продемонстрируйте работу.

Задание 5. Комплексная задача: "Библиотечный каталог" (10 баллов)
Создайте структуру `Book` с полями: автор, название, год, количество экземпляров.
Реализуйте меню:
1. Добавить книгу (ввод с клавиатуры, динамический массив).
2. Показать весь каталог.
3. Найти книги по автору (вывести все найденные).
4. Сохранить каталог в бинарный файл.
5. Загрузить каталог из бинарного файла.
6. Выход.
При сохранении в бинарный файл сохраняйте структуры напрямую с помощью `write`. При загрузке — читайте `read`.

Примеры индивидуальных заданий по курсу:

Задание №1. Калькулятор материалов для реставрации
Реставратору необходимо рассчитать количество краски для восстановления фрески. Напишите программу, которая запрашивает:
- Длину и ширину стены (в метрах)
- Расход краски на 1 кв.м (в граммах)
- Объем одной банки (в литрах, 1 литр = 1000 грамм)
Программа должна вывести:
- Площадь стены
- Необходимое количество краски в граммах
- Количество банок, которое нужно купить (округлить **вверх**)

Задание №2. Хронология музейных экспонатов

В музее хранятся экспонаты разных эпох. Напишите программу, которая запрашивает год создания экспоната (например, -550 для 550 года до н.э., 1945 для нашей эры). Программа должна определить и вывести:
- Век (с обозначением "до н.э." или "н.э.")
- Сколько лет прошло от этого события до текущего года (текущий год ввести с клавиатуры).

Задание №3. Билеты в театр
Театральная касса продает билеты. Стоимость билета зависит от ряда (ближе к сцене — дороже). Напишите программу, которая запрашивает:
- Стоимость билета в 1-м ряду
- Количество рядов
- На сколько процентов дешевле каждый следующий ряд (равномерное снижение)
Программа должна рассчитать и вывести стоимость билета для каждого ряда в виде таблицы.

Задание №4. Конвертер валют для международной выставки
Организаторы выставки закупают материалы за рубежом. Напишите программу-конвертер, которая переводит рубли в евро, доллары и юани. Курсы вводятся пользователем. Программа должна работать до тех пор, пока пользователь не введет 0 вместо суммы.

Задание №5. Допуск к экзамену
Студент допускается к экзамену, если:
- Посетил не менее 80% занятий
- Сдал все 3 лабораторные работы (оценка не ниже 4)
- Написал промежуточный тест на проходной балл (вводится отдельно)
Напишите программу, которая запрашивает необходимые данные и выводит "Допущен" или "Недопущен" с указанием причины.

Задание №6. Классификация произведений искусства
Программа запрашивает у пользователя:
- Жанр (живопись, скульптура, графика)
- Материал (масло, мрамор, бронза, бумага)
- Год создания
На основе этих данных программа должна отнести произведение к эпохе/стилю (Античность, Средневековье, Возрождение, Барокко, Современность) или выдать сообщение "Не удалось классифицировать".

Задание №7. Золотое сечение
Напишите программу, которая проверяет, соответствуют ли размеры прямоугольника (длина и ширина) правилу "золотого сечения". Отношение длины к ширине должно быть приблизительно равно 1.618. Задайте допустимую погрешность (например, 0.05). Используйте тернарный оператор для вывода результата.

Задание №8. Даты проведения фестиваля
Фестиваль проводится с 1 по 15 июня. Напишите программу, которая запрашивает день и месяц (числа) и проверяет, попадает ли эта дата в период проведения фестиваля. Учитывайте, что пользователь может ввести месяц как число (6) или как строку ("июнь"). Используйте `switch` для месяцев.

Задание №9. Расписание занятий

Напишите программу, которая выводит расписание звонков. Вводятся: время начала первого занятия (часы и минуты), длительность пары (минуты), длительность перерыва (минуты), количество пар. Программа должна вывести таблицу:

Пара	Начало	Конец
1	9:00	10:30
2	10:40	12:10
…	…	…

Задание №10. Анализ оценок группы
В группе 10 студентов. Напишите программу, которая запрашивает оценки каждого студента за экзамен (по 5-балльной шкале). Программа должна вывести:
- Средний балл по группе
- Количество отличников (оценка 5)
- Количество неуспевающих (оценка 2)
- Самую частую оценку (моду) в группе

Задание №11. Игра "Угадай число" (с историей)
Компьютер загадывает год исторического события (например, 988 — Крещение Руси). Пользователь пытается угадать. Программа дает подсказки "больше", "меньше", а также исторический факт, если пользователь сильно отклоняется (например, "Это было раньше, чем Куликовская битва"). После угадывания выводится количество попыток и историческая справка о событии.

Задание №12. Таблица умножения для младших классов
Напишите программу, которая генерирует и выводит таблицу умножения для заданного числа N (от 1 до 10) в виде:
N x 1 = N
N x 2 = 2N
...
N x 10 = 10N
Дополнительно: программа должна предложить пользователю проверить свои знания, задавая случайные примеры из этой таблицы и считая процент правильных ответов.

Задание №13. Статистика текста
Дан текст (массив символов), введенный пользователем. Напишите программу, которая подсчитывает:
- Общее количество символов (с пробелами)
- Количество слов
- Количество предложений (заканчиваются на '.', '!', '?')
- Самую часто встречающуюся букву (без учета регистра)

Задание №14. Сортировка имен
Введите с клавиатуры список из 10 фамилий (русских или английских). Реализуйте:
1. Сортировку методом "пузырька" в алфавитном порядке (от А до Я).
2. Вывод отсортированного списка.
3. Поиск фамилии, введенной пользователем, в отсортированном списке (бинарный поиск).

Задание №15. Матрица смежности для музея
В музее 5 залов. Напишите программу, которая создает матрицу смежности (двумерный массив 5x5), описывающую наличие переходов между залами (1 — есть переход, 0 — нет).

Заполнить матрицу случайно (но сделать так, чтобы музей был связным, хотя бы минимально). Программа должна вывести:
- Саму матрицу
- Определить, из какого зала больше всего выходов
- Есть ли залы, из которых нет выхода (тупиковые)

Задание №16. Обработка музыкального плейлиста
Создайте массив структур `Song` (название, исполнитель, длительность в секундах). Заполните массив данными (минимум 5 песен). Реализуйте функции:
- Вывести весь плейлист
- Найти самую длинную песню
- Посчитать общую длительность плейлиста (в минутах и секундах)
- Отсортировать песни по названию

Задание №17. Библиотека математических функций для искусствоведа
Создайте набор функций для расчетов, полезных в искусстве и дизайне:
- `goldenRatio(double a)` — вычисляет золотое сечение (a * 1.618)
- `colorChannel(int brightness)` — нормализует яркость цвета (приводит в диапазон 0-255)
- `perspective(double distance, double size)` — вычисляет видимый размер объекта на расстоянии
В `main()` продемонстрируйте работу всех функций на тестовых данных.

Задание №18. Рекурсивный обход каталогов (моделирование)
Создайте структуру `File` (имя, размер). Напишите рекурсивную функцию, которая "обходит" вложенную структуру папок (можно смоделировать вложенными массивами) и подсчитывает суммарный размер всех файлов. Функция должна выводить путь к каждой папке.

Задание №19. Ханойские башни (визуализация)
Напишите программу, которая решает головоломку "Ханойские башни" для N дисков (N вводит пользователь). Программа должна выводить последовательность перемещений в формате:
Переместить диск с 1 на 3
Переместить диск с 1 на 2
...
Подсказка: классическая рекурсивная задача.

Задание №20. Динамическая картотека
Создайте динамический массив структур `Student` (ФИО, группа, рейтинг). Количество студентов заранее неизвестно. Реализуйте:
1. Добавление нового студента (массив расширяется).
2. Удаление студента по фамилии (массив сжимается, память перевыделяется).
3. Вывод всего списка.
4. Поиск студента с максимальным рейтингом.

Задание №21. Функция с переменным числом параметров
Напишите функцию `findAverage`, которая принимает количество оценок и через указатели/переменное число параметров принимает сами оценки, а возвращает средний балл. Используйте механизм `va_list` (из библиотеки `cstdarg`) или просто передачу массива через указатель с размером.

Задание №22. Электронный дневник

Создайте программу для ведения журнала успеваемости. Информация о студентах хранится в файле `students.txt`. Программа должна уметь:
- Загружать данные из файла
- Добавлять нового студента
- Ставить оценку существующему студенту (по фамилии и предмету)
- Выводить список всех студентов
- Сохранять изменения в файл
Формат файла: каждая строка — "Фамилия;Группа;Оценка1;Оценка2;Оценка3"

Задание №23. Шифрование текстового файла
Напишите программу, которая шифрует текстовый файл с помощью шифра Цезаря (сдвиг каждой буквы на K позиций в алфавите). Программа должна:
- Запрашивать имя файла для шифрования
- Запрашивать ключ сдвига K
- Запрашивать режим: шифрование или дешифрование
- Сохранять результат в новый файл (например, `encrypted.txt`)
Учесть только русские или только английские буквы, остальные символы оставлять без изменений.

Задание №24. База данных "Музейные экспонаты"
Разработайте программу для учета музейных экспонатов. Каждый экспонат описывается структурой: название, автор, год создания, зал хранения, стоимость. Данные хранятся в бинарном файле. Реализуйте:
- Добавление нового экспоната
- Просмотр всех экспонатов
- Поиск экспонатов по залу
- Поиск самого дорогого экспоната
- Удаление экспоната по названию (перезапись файла без удаляемого элемента)

Задание №25. Итоговый проект: "Генератор кроссвордов" (усложненное)
Напишите программу, которая:
1. Читает из файла `words.txt` список слов (существительные в именительном падеже).
2. Размещает их в сетке кроссворда (матрице символов) по горизонтали и вертикали так, чтобы они пересекались (хотя бы частично).
3. Выводит получившуюся сетку на экран.
4. Сохраняет сетку в файл `crossword.txt`.
Это задание требует серьезного подхода к алгоритмам. Можно упростить: размещать слова только по горизонтали, но с общими буквами.

ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К ЗАЧЕТУ:

1. Состав языка C++. Алфавит языка. Служебные (ключевые) слова. Знаки операций. Разделители. Идентификаторы.
2. Структура программы на языке C++. Директивы препроцессора. Глобальные объявления. Функция `main()`. Блоки операторов.
3. Переменные и идентификаторы. Правила создания идентификаторов. Понятие переменной. Объявление переменных. Область видимости переменных.
4. Типы данных языка C++. Целочисленные типы (`int`, `short`, `long`, `long long`). Символьный тип (`char`). Вещественные типы (`float`, `double`). Логический тип (`bool`). Размеры типов (оператор `sizeof`).

5. Описание констант и переменных. Инициализация переменных. Литералы (целые, вещественные, символьные, строковые). Константы (`const`). Способы инициализации переменных (присваивание, прямая инициализация, универсальная инициализация `{}`).
6. Основные операции языка C++. Арифметические операции. Операции сравнения. Логические операции. Операции присваивания. Приоритет операций.
7. Директива препроцессора `#include`. Назначение. Формы записи (с угловыми скобками и кавычками). Поиск подключаемых файлов.
8. Ввод-вывод с использованием библиотеки `iostream`, `iomanip`. Потоки `cin`, `cout`, `cerr`, `clog`. Операторы `<<` и `>>`. Форматирование вывода: манипуляторы `endl`, `setw()`, `setprecision()`, `fixed`, `left`, `right`.
9. Директива препроцессора `#define`. Определение макроконстант. Макроопределения с параметрами (опасность использования, отличие от `const` и функций).
10. Библиотека математических функций `math.h` / `cmath`. Основные функции: `sqrt()`, `pow()`, `sin()`, `cos()`, `tan()`, `exp()`, `log()`, `fabs()`. Подключение библиотеки.
11. Условный оператор `if`. Полная и сокращенная форма. Вложенные условные операторы. Проблема "висячего `else`". Использование составных операторов (блоков `{}`).
12. Множественный выбор: оператор `switch`. Синтаксис. Структура `case`-меток. Использование `break` для предотвращения "проваливания". Ключевое слово `default`.
13. Инструкции перехода (`goto`, `continue`, `return`, `break`). Назначение и особенности применения. `break` в циклах и `switch`. `continue` в циклах. `return` для выхода из функции. Критика использования `goto`.
14. Оператор цикла с предусловием (`while`). Синтаксис. Особенности выполнения (возможно, ни одного выполнения). Типичные ошибки (отсутствие изменения условия).
15. Оператор цикла с постусловием (`do-while`). Синтаксис. Отличие от `while` (гарантированное выполнение хотя бы один раз). Области применения.
16. Решение задач циклической структуры. Табулирование функций (вывод значений функции на интервале с заданным шагом). Вычисление сумм и произведений числовых рядов (конечных и бесконечных с заданной точностью).
17. Оператор цикла с заданным числом повторений (`for`). Синтаксис. Инициализация, условие, итерация. Множественные выражения в частях цикла. Вложенные циклы.
18. Одномерные массивы. Объявление. Инициализация (списком, частичная, автоматическая). Доступ к элементам по индексу. Ввод и вывод значений элементов.
19. Использование датчика случайных чисел. Функции `srand()` и `rand()`. Генерация псевдослучайных чисел в заданном диапазоне. Заполнение массивов случайными значениями.
20. Решение задач обработки числовых значений одномерных массивов.
Нахождение суммы и произведения всех элементов или элементов, удовлетворяющих условию.
Нахождение максимального (минимального) значения и его индекса.
Подсчет количества элементов по заданному критерию.
21. Решение задач переупорядочивания элементов в массиве (сортировка).
Метод "пузырька" (bubble sort).
Метод прямого выбора (selection sort).
Понятие об устойчивости сортировки и вычислительной сложности (O-большое).
22. Двумерные массивы. Объявление и инициализация. Интерпретация как матрицы. Ввод значений (по строкам). Вывод в общепринятом табличном виде.
23. Решение задач обработки двумерных массивов. Обработка по строкам. Обработка по столбцам. Поиск элементов на главной и побочной диагоналях. Транспонирование матриц.
24. Строки в стиле C (массивы символов). Объявление и инициализация. Строковые литералы. Функции библиотеки `cstring` / `string.h`: `strlen()`, `strcpy()`, `strcat()`, `strcmp()`.
25. Решение задач обработки символьных строк. Подсчет символов. Поиск подстроки. Замена символов. Проверка на палиндром.

26. Определение и вызов пользовательских функций. Синтаксис определения. Тип возвращаемого значения. Тело функции. Механизм вызова и возврата управления.
27. Понятие прототипа функции. Назначение (объявление функции до ее определения). Синтаксис прототипа. Расположение прототипов (в заголовочных файлах или в начале программы).
28. Формальные и фактические параметры функции. Формальные параметры в определении. Фактические параметры при вызове. Соответствие по типу и количеству.
29. Понятие рекурсии. Рекурсивные алгоритмы. Прямая и косвенная рекурсия. Базовый случай (условие выхода). Рекурсивный шаг. Примеры: факториал, числа Фибоначчи. Достоинства и недостатки (наглядность vs. расход памяти).
30. Объявление, инициализация указателей, операции с указателями. Операторы `*` (разыменование) и `&` (взятие адреса). Нулевой указатель (`nullptr`). Арифметика указателей.
31. Связь массивов и указателей. Имя массива как константный указатель на первый элемент. Доступ к элементам через указатель (индексация и адресная арифметика).
32. Передача массива в функцию. Особенности передачи (всегда по адресу). Необходимость передачи размера массива отдельным параметром.
33. Решение задач обработки массивов через указатели. Итерация по массиву с помощью указателя. Использование указателей в функциях для модификации массивов.
34. Ссылки. Передача аргументов функции по ссылке. Объявление ссылки. Отличие ссылки от указателя. Преимущества передачи по ссылке (избежание копирования, возможность изменения). Константные ссылки.
35. Методы распределения динамической памяти. Динамические массивы. Статическая, автоматическая и динамическая память. Операторы `new` и `delete` для одиночных объектов. Операторы `new[]` и `delete[]` для массивов. Создание динамических массивов (одномерных и двумерных). Утечки памяти.
36. Перечисляемый тип (`enum` и `enum class`). Объявление. Значения элементов. Преимущества `enum class` (строгая типизация).
37. Объявления `typedef`. Создание псевдонимов для существующих типов. Использование `using` (C++11) как альтернативы `typedef`.
38. Объявление структурного шаблона и структурной переменной. Инициализация. Ключевое слово `struct`. Поля структуры. Создание переменных структурного типа. Инициализация (агрегатная, через конструктор по умолчанию).
39. Работа со структурами через указатели. Оператор `->` для доступа к полям. Передача структур в функцию по указателю.
40. Объединения (`union`). Особенности распределения памяти (все поля разделяют одну область памяти). Размер объединения. Практическое применение.
41. Решение задач обработки структурированных данных. Создание массива структур. Поиск, сортировка, добавление и удаление записей в массиве структур.
42. Текстовые и бинарные файлы. Понятие файла. Отличия текстового и бинарного форматов хранения. Преобразования при записи/чтении (в текстовом режиме).
43. Функции работы с текстовыми файлами. Классы `ifstream`, `ofstream`, `fstream`. Методы `open()`, `close()`, `is_open()`. Чтение по словам, по строкам (`getline()`), посимвольно (`get()`). Запись (`<<`).
44. Функции работы с бинарными файлами. Режимы открытия: `ios::binary`. Методы `read()` и `write()`. Прямой доступ: `seekg()`, `seekp()`, `tellg()`, `tellp()`.
45. Решение задач обработки данных, организованных в файлы. Сохранение массива структур в файл (текстовый и бинарный). Загрузка данных из файла. Поиск записи в файле. Модификация данных в файле.

ПРИМЕРНЫЕ ВОПРОСЫ К ЗАЧЁТУ:

1. Состав языка C++
2. Структура программы языка C++
3. Переменные, идентификаторы
4. Типы данных языка C++
5. Описание констант и переменных. Инициализация переменных
6. Основные операции языка C++
7. Директива препроцессора #include
8. Ввод-вывод с использованием библиотеки iostream, iomanip
9. Директива препроцессора #define
10. Библиотека математических функций math.h
11. Условный оператор if
12. Множественный выбор: оператор switch
13. Инструкции перехода (goto, continue, return, break)
14. Оператор цикла с предусловием
15. Оператор цикла с постусловием
16. Решение задач циклической структуры (подсчет сумму(произведения) значений числового ряда, вывод значений функции на интервале)
17. Оператор цикла с заданным числом повторений
18. Одномерные массивы (объявление, инициализация, задание значений)
19. Использование счетчика случайных чисел для задания значений переменных и массивов
20. Решение задач обработки числовых значений одномерных массивов (нахождение суммы (произведения) всех элементов массива или частично (по некоторому условию), нахождение значения максимума (минимума) из всех элементов или по некоторому условию)
21. Решение задач переупорядочивания элементов в массиве (методы сортировки «пузырька», метод прямого выбора)
22. Двумерные массивы (объявление, инициализация, ввод значений, вывод в общепринятом виде)
23. Решение задач обработки двумерных массивов по строкам и по столбцам
24. Строки (объявление, инициализация). Функции работы со строками библиотеки string.h
25. Решение задач обработки символьных строк
26. Определение, вызов пользовательских функций
27. Понятие прототипа функции
28. Формальные и фактические параметры функции
29. Понятие рекурсии. Рекурсивные алгоритмы
30. Объявление, инициализация указателей, операции с указателями
31. Связь массивов и указателей
32. Передача массива в функцию
33. Решения задач обработки массивов способом обращения к значениям через указатели
34. Ссылки. Передача аргументов функции по ссылке
35. Методы распределения динамической памяти. Динамические массивы
36. Перечисляемый тип
37. Объявления typedef
38. Объявление структурного шаблона и структурной переменной, инициализация
39. Работа со структурами через указатели
40. Объединения
41. Решение задач обработки структурированных данных
42. Текстовые и бинарные файлы
43. Функции работы с текстовыми файлами

44. Функции работы с бинарными файлами
45. Решение задач обработки данных, организованных в файлы

ПРИМЕРНЫЕ ТЕМЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ (ЛАБОРАТОРНЫЕ РАБОТЫ ВЫПОЛНЯЮТСЯ В ИНДИВИДУАЛЬНОМ ГРАФИКЕ)

Лабораторная работа № 1 Общие замечания

Процесс создания программы в C++Builder состоит из двух шагов: сначала нужно создать форму программы (диалоговое окно), а затем функции обработки событий. Форма приложения Windows создается путем добавления в нее компонентов и последующей их. В форме практически любого приложения есть компоненты, которые обеспечивают интерфейс между программой и пользователем. Такие компоненты называют базовыми. К базовым компонентам относятся:

Label - поле вывода текста; Edit - поле редактирования текста; Button - командная кнопка; CheckBox - независимая кнопка выбора; RadioButton - зависимая кнопка выбора; ListBox - список выбора; ComboBox - комбинированный список выбора.

Вид компонента, его размер и поведение определяют значения (характеристик) компонента.

Основную работу в программе выполняют функции обработки событий.

Исходную информацию программа может получить из полей редактирования (компонент Edit), списка выбора (компонент ListBox) или комбинированного списка (компонент ComboBox).

Для ввода значений логического типа можно использовать Компоненты CheckBox и RadoiButton.

Результат программа может вывести в поле вывода текста (компонент Label) или в окно сообщения (функции ShowMessage, MessageDlg).

Для преобразования текста, например, находящегося в поле редактирования, в целое число нужно использовать функцию StrToint, а в дробное - функцию StrToFloat. Для преобразования целого, например, значения переменной, в строку нужно использовать функцию IntTostr, а для преобразования дробного - функцию FloatToStr или FloatToStrF.

Создадим приложение, которое при нажатии кнопки перемножает два числа, введенных пользователем, и показывает результат умножения. Эти числа будем понимать, как длину и ширину сторон прямоугольника, и тогда результат - это площадь.

При построении этого приложения используйте компоненты - окна редактирования LabeledEdit. Результат нужно выводить не в метку Label, а в панель Panel, чтобы испытать:

1 Откройте новое приложение.

2.Перенесите на форму со страницы библиотеки Additional два окна компонента LabeledEdit, а со страницы библиотеки Standard - одну панель Panel, одну кнопку Button и одну метку Label для надписи.

3 Измените надписи в метках компонентов LabeledEdit на «Ширина», «Высота». Для этого щелкните на символе "+" в свойстве EditLabel этих компонентов и измените надпись в свойстве Caption раскрывшихся списков свойств меток. Задайте для меток жирный шрифт.

4 Измените свойство Caption кнопки на «Расчет». Очистите свойство Caption у панели.

В свойстве Text компонентов LabeledEdit задайте 1" - начальное значение текста. Установите свойства BevelInner = bvLowered и BevelOuter = bvRaised панели, которые определяют вид (утопленный - bvLowered или выпуклый bvRaised) основного поля и рамки панели.

Напишите обработчик щелчка кнопки. Операторы этого обработчика имеют вид:

```
Label1->Caption=”Площадь прямоугольника равна:”;
Panel1->Caption = LabeledEdit1->Text + " * " + LabeledEdit2->Text + " = " +
FloatToStr(StrToFloat(LabeledEdit1->Text) * StrToFloat(LabeledEdit2->Text));
```

Во втором операторе свойству Caption компонента Panel1 присваивается значение выражения, указанного в правой части оператора. Это выражение должно иметь тип строки текста. Начинается строка с текста, введенного пользователем в окно редактирования Edit1 - этот текст хранится в свойстве Text. Затем прибавляете к этому тексту символы " * ". Знак "+" в выражениях для строк означает конкатенацию - сцепление двух строк символов. Затем аналогичным образом к строке добавляется текст второго окна редактирования и символы " = ". После вставляется результат перемножения двух целых чисел. Этот результат будет числом и, чтобы вставить его в текст, надо сначала преобразовать это число в строку. Эту операцию выполняет функция FloatToStr(...), которая преобразует в строку само произведение двух чисел. Но числа заданы пользователем в виде текстов - строк символов в окнах редактирования. Прежде, чем перемножать, эти строки надо перевести в числа. Эту операцию выполняют функции StrToFIoat(), преобразующие символьное изображение числа в его значение типа действительного числа. Знак '*', указанный между двумя функциями StrToFIoat, обозначает операцию умножения.

Лабораторная работа 2. Введение в среду разработки и базовый ввод/вывод. (Структура программы, типы данных)

Задание: Разработать программу-"визитку". Запросить у пользователя ФИО, год рождения, группу. Вывести форматированную карточку студента.

Лабораторная работа 3. Программирование линейных алгоритмов. (Арифметика, математические функции)

Задание: Вычисление площади и периметра геометрических фигур. Перевод единиц измерения информации (биты в байты, килобайты и т.д.).

Лабораторная работа 4. Логические выражения и условный оператор. (if/else)

Задание: Решение квадратного уравнения с анализом дискриминанта. Определение високосного года.

Лабораторная работа 5. Множественный выбор. (Switch-case)

Задание: Создать "Электронное меню столовой": пользователь вводит номер блюда, программа выводит его название и цену. Обработка ошибочного ввода.

Лабораторная работа 6. Организация циклов с параметром. (for)

Задание: Вычисление суммы ряда, факториала числа. Табулирование функции (вывод таблицы значений синуса/косинуса на интервале).

Лабораторная работа 7. Организация циклов с условием. (while / do-while)

Задание: Программа "Угадай число" (компьютер загадывает, пользователь угадывает с подсказками "больше/меньше"). Обработка ввода до тех пор, пока пользователь не введет корректное значение.

Лабораторная работа 8. Итерационные алгоритмы. Вычисление сумм и произведений рядов.

Задание: Вычисление числа Пи или экспоненты с заданной точностью (разложение в ряд Тейлора).

Лабораторная работа 9. Одномерные массивы: поиск и замена элементов.

Задание: Ввод массива оценок студента. Найти средний балл, максимальную и минимальную оценку, подсчитать количество долгов (оценок ниже 4).

Лабораторная работа 10. Одномерные массивы: сортировка данных.

Задание: Реализовать сортировку массива фамилий в алфавитном порядке (методом "пузырька" или выбора).

Лабораторная работа 11. Двумерные массивы: обработка матриц.

Задание: Работа с таблицей данных (например, расписание занятий: дни недели / номера пар). Поиск "окон" в расписании. Вычисление суммы элементов по диагоналям.

Лабораторная работа 12. Многомерные массивы. Поиск в неоднородных структурах.

Задание: Хранение данных о нескольких группах студентов (группа -> список студентов). Вывод студентов, имеющих задолженности по предметам.

Лабораторная работа 13. Функции: способы передачи параметров.

Задание: Написать набор функций для работы с обыкновенными дробями (сложение, умножение, сокращение). Передача параметров по значению и по ссылке.

Лабораторная работа 14. Рекурсивные функции.

Задание: Обход файловой системы (теоретически). Алгоритмы быстрой сортировки. Ханойские башни (классическая задача).

Лабораторная работа 15. Указатели и адресная арифметика.

Задание: Написать функции, которые меняют местами значения двух переменных с использованием указателей. Реализовать собственную функцию копирования строк (аналог strcpy).

Лабораторная работа 16. Динамическое выделение памяти под массивы.

Задание: Создать программу, которая запрашивает у пользователя количество элементов, создает динамический массив, заполняет его и находит сумму элементов. Важно корректно освободить память.

Лабораторная работа 17. Работа с динамическими строками (массивы символов).

Задание: Подсчет количества слов в предложении, замена символов. Реализация без использования библиотеки string (только cstring или через индексы).

Лабораторная работа 18. Структуры: создание и базовые операции.

Задание: Создать структуру "Студент" (ФИО, номер группы, рейтинг). Ввести данные о нескольких студентах и вывести список отсортированных по рейтингу.

Лабораторная работа 19. Массивы структур. Обработка баз данных в памяти.

Задание: Создать базу данных "Библиотечный каталог" (структура "Книга": автор, название, год, наличие). Реализовать поиск книги по автору.

Лабораторная работа 20. Объединения и перечисления. Битовая экономия памяти.

Задание: Использовать перечисление enum для дней недели или месяцев. Использовать объединение для хранения разных типов данных (int/float) в одном поле (на практике показать, как это работает).

Лабораторная работа 21. Динамические массивы структур.

Задание: Система учета студентов. Количество студентов заранее неизвестно. Программа просит вводить данные, пока пользователь не введет "стоп". Хранение в динамически расширяемом массиве.

Лабораторная работа 22. Текстовые файлы: запись и чтение.

Задание: Сохранить данные структуры "Студент" в текстовый файл. Затем прочитать их из файла и вывести на экран в виде таблицы.

Лабораторная работа 23. Бинарные файлы: прямые операции с данными.

Задание: Сохранить массив структур в бинарный файл. Прочитать из файла данные о конкретном студенте по его порядковому номеру (прямой доступ).

Лабораторная работа 24. Обработка файлов сложной структуры.

Задание: Дан текстовый файл с неизвестным количеством чисел, разделенных пробелами. Прочитать все числа в динамический массив и вычислить их среднее арифметическое.

Лабораторная работа 25. Комплексная работа: База данных "Студенческая группа".

Задание: Объединить структуры и файлы. Реализовать консольное приложение: добавление записей в файл, просмотр всех записей, поиск по фамилии (с выводом результата), сортировка записей в файле.

Примерные вопросы для семинарских занятий:

1. Алгоритм и его свойства. Блок-схемы. Разбор понятия алгоритма. Составление блок-схем для линейных и разветвляющихся алгоритмов (до написания кода).
Вопросы для подготовки к семинару 1:

1. Понятие алгоритма. Интуитивное и формальное определение.

2. Основные свойства алгоритмов.

3. Способы описания алгоритмов.

4. Графические элементы блок-схем (ГОСТ 19.701-90).

5. Построение линейных алгоритмов.

6. Построение разветвляющихся алгоритмов.

2. Структура программы на C++. Препроцессорная обработка. Детальный разбор директивы `#include`, функции `main()`. Что такое пространство имен `std`. Разбор типичных ошибок линковки.
3. Базовые типы данных и модификаторы. Обсуждение памяти: сколько байт занимают `int`, `char`, `float`. Переполнение переменных. Спецификаторы `short`, `long`, `unsigned`.
4. Семинар 4: Форматированный ввод/вывод. Разбор возможностей `cout` и `cin`. Манипуляторы (`endl`, `hex`, `dec`, `setw`, `setprecision` из `<iomanip>`). Потоковый ввод/вывод.
5. Контрольная работа №1: "Линейные алгоритмы и типы данных". Проверка умения писать простые программы и правильно выбирать типы данных.
6. Логические операции и выражения. Таблицы истинности для `&&`, `||`, `!`. Приоритеты операций. Сложные логические условия.
7. Оператор ветвления `if-else`. Разбор полетов. Анализ сложных кейсов: висячий `else`, вложенные условия. Способы написания читаемого кода.
8. Тернарный оператор и оператор `switch`. Когда уместен `?:`. Сравнение `switch` и длинной цепочки `if-else`. Проваливание (`fallthrough`) в `switch`.
9. Обработка ошибок ввода. Как защитить программу от дурака. Проверка корректности ввода данных пользователем (`cin.fail()`, `cin.clear()`).
10. Практикум по решению задач на ветвление. Решение задач повышенной сложности (например, классификация треугольников по сторонам, расчет даты Пасхи).
11. Итерация и рекуррентные последовательности. Разбор понятия "итерация". Вычисление членов последовательности через предыдущие значения.
12. Цикл `for`: тонкости использования. Нестандартное применение цикла: несколько переменных, пустое тело, вечный цикл.
13. Циклы `while` и `do-while`. Различия и применение. Обсуждение, когда нужен постусловие, а когда предусловие. Типичные ошибки с забытым инкрементом.
14. Вложенные циклы. Табличная печать. Анализ работы вложенных циклов на примере печати узоров (пирамидок, ромбов) из символов '*'.
15. Управление циклами: `break` и `continue`. Разбор спорных моментов использования операторов прерывания. Читаемость кода.
16. Контрольная работа №2: "Циклические алгоритмы". Проверка навыков работы с циклами и вложенными структурами.
17. Массивы: представление в памяти. Как массив хранится в памяти. Индексация. Почему индексы с 0. Выход за границы массива (UB — неопределенное поведение).
18. Алгоритмы поиска в массиве. Линейный поиск. Поиск максимума/минимума. Анализ эффективности (количество сравнений).
19. Сортировка массивов: метод пузырька. Детальный разбор алгоритма сортировки пузырьком. Ручная трассировка. Модификации с флагом обмена.

20. Сортировка массивов: метод выбора и вставками. Сравнение алгоритмов сортировки. Понятие о вычислительной сложности (О-большое) на интуитивном уровне.
21. Двумерные массивы: обработка строк и столбцов. Алгоритмы работы с матрицами. Вычисление сумм по строкам/столбцам. Транспонирование.
22. Семинар-игра: "Морской бой" (логика расстановки). Разбор представления игрового поля в виде двумерного массива. Проверка корректности расстановки кораблей.
23. Функции: область видимости и время жизни переменных. Локальные, глобальные и статические переменные. Стек вызовов функций.
24. Способы передачи параметров. Разбор передачи по значению, по указателю и по ссылке. Что выбрать и почему. Константные ссылки.
25. Возврат значений из функции. Возврат нескольких значений. Способы вернуть более одного результата (через ссылки/указатели, через структуру).
26. Перегрузка функций. Поиск ошибок неоднозначности. Как компилятор выбирает нужную функцию. Неоднозначные вызовы.
27. Рекурсия: стек вызовов и базовый случай. Детальный разбор работы рекурсии на примере факториала и чисел Фибоначчи. Риск переполнения стека.
28. Практикум: Разработка библиотеки функций. Создание модуля для работы с массивами (свои функции сортировки, печати, заполнения). Разделение на `.h` и `.cpp`.
29. Указатели: адресная арифметика. Связь указателей и массивов. Разбор операций `++`, `--`, `+` над указателями.
30. Указатели и функции. Передача массивов. Почему массив всегда передается по адресу. Функции, изменяющие исходный массив.
31. Динамическая память: `new` и `delete`. Разбор кучи (heap) и стека (stack). Утечки памяти. Правило "когда выделил — освободи".
32. Динамические массивы. Создание двумерных динамических массивов. Освобождение памяти.
33. Умные указатели (обзор C++11 и далее). Понятие об `unique_ptr` и `shared_ptr`. Зачем они нужны. RAII (как концепция).
34. Структуры: выравнивание и упаковка. Как структура лежит в памяти. Понятие выравнивания (`#pragma pack`). Размер структуры.
35. Массивы структур как простейшие базы данных. Реализация CRUD-операций (создание, чтение, обновление, удаление) над массивом структур.
36. Работа с текстовыми файлами. Открытие, закрытие, проверка открытия. Различные режимы работы. Разбор ошибок работы с файлами.
37. Работа с бинарными файлами. Структуры и файлы. Чтение и запись структур в бинарном виде. Прямой доступ к записям (`seekg`, `tellg`).
38. Итоговое коллоквиум. Защита проектов. Собеседование по ключевым темам курса. Презентация и защита итоговых лабораторных работ (проектов).

Примерные варианты тестов:
ТЕСТ №1
Выберите один правильный вариант ответа из предложенных.

1. Что обязательно должна содержать любая исполняемая программа на C++?
A) Директиву `#include <iostream>`
B) Функцию `main()`
C) Библиотеку `math.h`
D) Оператор `return 0;`

2. Для чего используется директива `#include <iostream>`?
A) Для подключения математических функций
B) Для подключения средств ввода/вывода

C) Для объявления пространства имен
D) Для завершения программы

3. Какое пространство имен содержит стандартные функции C++?
A) `system`
B) `cpp`
C) `std`
D) `iostream`

4. Что означает строка `using namespace std;`?
A) Создание нового пространства имен
B) Подключение стандартной библиотеки
C) Объявление о том, что в коде будут использоваться имена из пространства имен `std`
D) Завершение программы

5. Какой оператор используется для проверки нескольких условий в C++?
A) `if-else`
B) `for`
C) `while`
D) `goto`

6. Что выведет следующий фрагмент кода?
```
int x = 5;
if (x = 10)
   cout << "Равно";
else
   cout << "Не равно";
```
A) Равно
B) Не равно
C) Ошибка компиляции
D) Ничего не выведет

7. Какое логическое выражение соответствует утверждению "x принадлежит интервалу (0, 10]"?
A) `x > 0 && x < 10`
B) `x > 0 && x <= 10`
C) `x >= 0 && x <= 10`
D) `x > 0 || x <= 10`

8. В каком случае используется оператор `switch`?
A) Для организации циклов
B) Для выбора одного из нескольких вариантов по значению выражения
C) Для объявления переменных
D) Для обработки исключений

9. Что произойдет, если в операторе `switch` не использовать `break` в блоке `case`?
A) Произойдет ошибка компиляции
B) Будет выполнен только текущий блок
C) Программа завершится аварийно
D) Выполнение "провалится" дальше в следующий блок

10. Какой цикл гарантированно выполнится хотя бы один раз?

A) `for`
B) `while`
C) `do-while`
D) Все циклы могут не выполниться ни разу

11. Сколько раз выполнится тело цикла?
```
for (int i = 0; i < 5; i++) {
   // тело цикла
}
```
A) 4 раза
B) 5 раз
C) 6 раз
D) Бесконечное количество раз

12. Что произойдет при выполнении этого кода?
```
int i = 1;
while (i > 0) {
   i++;
}
```
A) Цикл выполнится 1 раз
B) Цикл выполнится 10 раз
C) Произойдет зацикливание (бесконечный цикл)
D) Ошибка компиляции из-за отсутствия условия

13. Как можно прервать выполнение цикла досрочно?
A) Использовать `continue`
B) Использовать `break`
C) Использовать `return`
D) Использовать `exit`

14. Что делает оператор `continue`?
A) Завершает выполнение цикла
B) Завершает выполнение программы
C) Переходит к следующей итерации цикла, пропуская оставшийся код
D) Ничего не делает

15. Что такое прототип функции?
A) Тело функции
B) Объявление функции, содержащее ее имя, тип возвращаемого значения и параметры
C) Вызов функции
D) Переменная внутри функции

16. Каким способом передаются параметры в функцию по умолчанию в C++?
A) По ссылке
B) По указателю
C) По значению
D) По адресу

17. Что означает ключевое слово `void` перед именем функции?
A) Функция не принимает параметров
B) Функция ничего не возвращает
C) Функция возвращает целое число

D) Функция является шаблонной

18. Что такое рекурсивная функция?
A) Функция, которая вызывает саму себя
B) Функция, которая вызывает другую функцию
C) Функция без параметров
D) Функция, возвращающая вещественное число

19. Что хранит указатель?
A) Значение переменной
B) Имя переменной
C) Адрес переменной в памяти
D) Тип переменной

20. Как правильно объявить указатель на целое число?
A) `int ptr;`
B) `int &ptr;`
C) `int* ptr;`
D) `pointer int ptr;`

21. Для чего используется оператор `new`?
A) Для создания новой переменной на стеке
B) Для выделения памяти в куче (динамической памяти)
C) Для удаления переменной
D) Для получения адреса переменной

22. Какой индекс имеет первый элемент массива `int arr[10];`?
A) 1
B) 0
C) -1
D) Любой

23. Что произойдет при обращении к элементу массива за его границами?
A) Ошибка компиляции
B) Автоматическое расширение массива
C) Неопределенное поведение (может быть ошибка, может быть чтение "мусора")
D) Программа всегда вылетит с сообщением об ошибке

24. Как называется пользовательский тип данных, объединяющий несколько переменных разных типов?
A) Массив
B) Структура (`struct`)
C) Указатель
D) Перечисление (`enum`)

25. Какой класс используется для записи в файл в C++?
A) `ifstream`
B) `ofstream`
C) `fstream`
D) `file`

ТЕСТ №2

Инструкция: Выберите один правильный вариант ответа из предложенных.

1. Какой символ используется для завершения большинства инструкций в C++?
A) `.` (точка)
B) `,` (запятая)
C) `;` (точка с запятой)
D) `:` (двоеточие)

2. Как правильно закомментировать несколько строк кода?
A) `// комментарий //`
B) `<!-- комментарий -->`
C) `/* комментарий */`
D) `# комментарий`

3. Что означает `endl` в C++?
A) Конец файла
B) Пустую строку
C) Переход на новую строку и сброс буфера
D) Очистку экрана

4. Какая функция возвращает код возврата программы операционной системе?
A) `main()`
B) `return`
C) `exit`
D) `system`

5. Какое значение будет иметь переменная `y` после выполнения кода?
```
int x = 7;
int y = (x > 5) ? 10 : 20;
```
A) 5
B) 7
C) 10
D) 20

6. Как проверить, что переменная `ch` является заглавной латинской буквой?
A) `ch >= 'A' && ch <= 'Z'`
B) `ch > 'A' && ch < 'Z'`
C) `ch = 'A' || ch = 'Z'`
D) `ch >= A && ch <= Z`

7. Сколько раз выполнится проверка условия в этом коде?
```
int a = 5, b = 3;
if (a > b)
   if (a > 0)
      cout << "a положительное и больше b";
```
A) 1 раз
B) 2 раза
C) Ни разу
D) Зависит от значений

8. Что выведет код?
```
int a = 10;
```

```
if (a % 2 == 0)
   cout << "Четное";
else
   cout << "Нечетное";
```
A) Четное
B) Нечетное
C) Ошибка: % не работает с целыми числами
D) Ничего

9. Можно ли в операторе `switch` использовать переменную типа `double`?
A) Да, всегда
B) Да, но с ограничениями
C) Нет, только целочисленные типы и `char`
D) Нет, только `int`

10. Чем отличается цикл `while` от `do-while`?
A) `while` проверяет условие после выполнения тела, `do-while` — до
B) `while` проверяет условие до выполнения тела, `do-while` — после
C) Ничем, это синонимы
D) `while` используется только с числами, `do-while` — с символами

11. Что выведет этот код?
```
for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 2; j++) {
      cout << "*";
   }
   cout << endl;
}
```
A) `**\n**\n**\n`
B) `***\n***\n`
C) `******\n`
D) `*\n*\n*\n`

12. Как создать бесконечный цикл с помощью `for`?
A) `for(;;)`
B) `for(infinite)`
C) `for(while)`
D) `for(1)`

13. Что произойдет при выполнении?
```
int i = 0;
while(i < 5) {
   if(i == 3) continue;
   cout << i;
   i++;
}
```
A) Выведется `0124`
B) Выведется `012`
C) Произойдет зацикливание, так как `i++` не выполнится при `i == 3`
D) Ошибка компиляции

14. Сколько раз выполнится цикл?

```
int i = 10;
do {
    i--;
} while(i > 10);
```
A) 0 раз
B) 1 раз
C) 10 раз
D) Бесконечно

15. Что такое перегрузка функций?
A) Создание нескольких функций с разными именами
B) Создание нескольких функций с одинаковым именем, но разными параметрами
C) Вызов функции внутри другой функции
D) Слишком большое количество функций в программе

16. Как передать переменную в функцию так, чтобы функция могла изменить её значение?
A) Передать по значению
B) Передать по ссылке или указателю
C) Сделать переменную глобальной
D) Варианты B и C верны

17. Что возвращает функция, если в ней нет оператора `return`?
A) 0
B) -1
C) Неопределенное значение (мусор)
D) Ошибка компиляции

18. Для чего нужны параметры по умолчанию в функции?
A) Для автоматического определения типа параметра
B) Чтобы можно было вызывать функцию без указания всех аргументов
C) Чтобы увеличить скорость выполнения
D) Чтобы скрыть параметры от других функций

19. Как получить адрес переменной `x`?
A) `*x`
B) `&x`
C) `@x`
D) `addr(x)`

20. Что произойдет при выполнении `delete` для одного и того же указателя дважды?
A) Корректное освобождение памяти
B) Ничего не произойдет
C) Неопределенное поведение (обычно программа аварийно завершается)
D) Второй `delete` игнорируется

21. Чем ссылка отличается от указателя?
A) Ссылка не может быть перепривязана к другой переменной
B) Ссылка занимает больше памяти
C) Ссылка не требует инициализации
D) Указатель нельзя изменить

22. Как объявить двумерный массив 3x4?

A) `int arr[3,4];`
B) `int arr[3][4];`
C) `int arr[12];`
D) `int[3][4] arr;`

23. Имя массива в C++ — это:
A) Указатель на первый элемент массива
B) Переменная, хранящая размер массива
C) Специальный тип данных
D) Ссылка на последний элемент

24. Как обратиться к полю структуры, если есть указатель на структуру `ptr`?
A) `ptr.field`
B) `ptr->field`
C) `ptr::field`
D) `ptr.*field`

25. Какой режим открытия файла нужен для добавления данных в конец файла?
A) `ios::in`
B) `ios::out`
C) `ios::app`
D) `ios::trunc`

ТЕСТ №3
Инструкция: Выберите один правильный вариант ответа из предложенных.

1. Какая библиотека содержит математические функции (sin, cos, sqrt)?
A) `<iostream>`
B) `<cmath>` или `<math.h>`
C) `<string>`
D) `<algorithm>`

2. Что произойдет, если не написать `return 0;` в конце функции `main()`?
A) Ошибка компиляции
B) Программа не запустится
C) Компилятор автоматически добавит `return 0;` (в C++99 и новее)
D) Программа вернет случайное число

3. Какой тип данных используется для хранения одного символа?
A) `string`
B) `char`
C) `symbol`
D) `int`

4. Как объявить константу в C++?
A) `const int MAX = 100;`
B) `constant int MAX = 100;`
C) `#define MAX 100`
D) Варианты A и C верны (хотя A — более правильный для C++)

5. Что такое "висячий else"?
A) Ситуация, когда else относится к ближайшему if

B) else без if
C) Ошибка компиляции
D) Специальный оператор

6. Какое условие нужно записать, чтобы проверить, что год `y` является високосным? (Високосный: делится на 4, но не на 100, кроме случаев, когда делится на 400)
A) `y % 4 == 0`
B) `(y % 4 == 0 && y % 100 != 0) || y % 400 == 0`
C) `y % 4 == 0 && y % 100 == 0 && y % 400 == 0`
D) `y % 400 == 0`

7. Что выведет код?
```
int a = 5;
int b = 10;
if (a > b)
   if (a > 0) cout << "A";
else cout << "B";
```
A) A
B) B
C) Ничего
D) Ошибка

8. Какой оператор проверяет равенство двух значений?
A) `=`
B) `= =`
C) `! =`
D) `= = =`

9. Можно ли в `switch` использовать несколько меток `case` для одного блока кода?
A) Нет, каждая метка должна иметь свой код
B) Да, можно написать `case 1: case 2: cout << "1 или 2"; break;`
C) Только если использовать `goto`
D) Да, но только для символьных констант

10. Какой цикл лучше всего подходит для перебора элементов массива, когда известно их количество?
A) `while`
B) `do-while`
C) `for`
D) Любой

11. Что выведет код?
```
int i = 0;
while(i < 5) {
   i++;
   if(i == 3) break;
   cout << i << " ";
}
```
A) `1 2 3`
B) `1 2`
C) `0 1 2`
D) `1 2 3 4 5`

12. Сколько раз выполнится цикл?
```
for(int i = 10; i > 0; i--);
```
A) 9 раз
B) 10 раз
C) 11 раз
D) Ни разу, так как после for сразу стоит точка с запятой

13. В какой части цикла `for` можно объявить переменную?
A) Только в инициализации
B) Только в условии
C) Только в итерации
D) В любой части

14. Что произойдет при выполнении этого кода?
```
int i = 0;
while(i < 5);
    i++;
```
A) Цикл выполнится 5 раз
B) Цикл выполнится 1 раз
C) Бесконечный цикл
D) Ошибка компиляции

15. Что такое сигнатура функции?
A) Тело функции
B) Имя функции и список параметров
C) Тип возвращаемого значения
D) Адрес функции в памяти

16. Может ли функция возвращать массив?
A) Да, всегда
B) Нет
C) Может вернуть указатель на массив
D) Только если массив глобальный

17. Что такое inline-функция?
A) Функция, определенная внутри другой функции
B) Функция, код которой подставляется непосредственно в место вызова для ускорения
C) Функция без имени
D) Функция, работающая с потоками ввода/вывода

18. Что будет, если вызвать рекурсивную функцию без базового случая (условия выхода)?
A) Ничего особенного
B) Функция выполнится один раз
C) Произойдет переполнение стека (stack overflow)
D) Компилятор исправит ошибку

19. Что такое "утечка памяти"?
A) Потеря данных при выключении компьютера
B) Ситуация, когда программа не освобождает выделенную память
C) Слишком быстрое заполнение памяти
D) Вирусная атака

20. Как правильно освободить память, выделенную под массив `int* arr = new int[10];`?
A) `delete arr;`
B) `free(arr);`
C) `delete[] arr;`
D) `delete arr[];`

21. Что выведет код?
int x = 5;
int& ref = x;
ref = 10;
cout << x;
A) 5
B) 10
C) Адрес x
D) Ошибка

22. Как передать двумерный массив в функцию?
A) `void func(int arr[][10], int rows)`
B) `void func(int** arr, int rows, int cols)`
C) `void func(int arr[10][10])`
D) Все варианты возможны в разных контекстах

23. Как вычислить количество элементов в статическом массиве `int arr[20];`?
A) `sizeof(arr)`
B) `sizeof(arr) / sizeof(arr[0])`
C) `length(arr)`
D) `arr.size()`

24. Чем структура (`struct`) отличается от объединения (`union`) в C++?
A) В `union` все поля хранятся одновременно, в `struct` — по очереди
B) В `union` все поля используют одну область памяти, в `struct` — каждый свою
C) Ничем, это синонимы
D) `union` нельзя использовать в функциях

25. Как проверить, успешно ли открылся файл?
A) `if (file)`
B) `if (file.is_open())`
C) `if (file.good())`
D) Все варианты верны

Критерии оценки: 90-100% — "отлично", 75-89% — "хорошо", 60-74% — "удовлетворительно", менее 60% — "неудовлетворительно".

СИСТЕМА ОЦЕНИВАНИЯ

Форма контроля	**Компетенция**	**Оценка**
Текущий контроль: *- участие в дискуссии на семинаре*	*ОПК-6* *ОПК-8*	*зачтено/не зачтено*

- выполнение лабораторных работ		*отлично/хорошо/удовлетворительно/неудовлетворительно*
Промежуточная аттестация *Зачет* *Зачет с оценкой*	*ОПК-1* *ОПК-8*	*зачтено /не зачтено* *зачтено (отлично, хорошо, удовлетворительно)/ не зачтено*

6.2. Критерии оценки результатов по дисциплине[1]

Оценка по дисциплине	**Критерии оценки результатов обучения по дисциплине**
«отлично»/ «зачтено»	Выставляется обучающемуся, если компетенция(ии), закрепленная за дисциплиной, сформирована (по индикаторам/ результатам обучения в формате знать-уметь-владеть) в полном объеме на уровне «высокий», и обучающийся демонстрирует как результат обучения следующие знания, умения и навыки: обучающийся глубоко и прочно усвоил теоретический и практический материал, продемонстрировал это на занятиях и в ходе промежуточной аттестации. Обучающийся исчерпывающе и логически стройно излагает учебный материал, умеет сочетать теорию с практикой, справляется с решением задач профессиональной направленности высокого уровня сложности, правильно обосновывает принятые решения. Свободно ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации.
«хорошо»/ «зачтено»	Выставляется обучающемуся, если он знает теоретический и практический материал, грамотно и по существу излагает его на занятиях и в ходе промежуточной аттестации, не допуская существенных неточностей. Обучающийся правильно применяет теоретические положения при решении практических задач профессиональной направленности разного уровня сложности, владеет необходимыми для этого навыками и приёмами. Достаточно хорошо ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне «хороший».

[1] Могут уточняться и дополняться в соответствии со спецификой дисциплины, установленных форм контроля, применяемых технологий обучения и оценивания.

Оценка по дисциплине	Критерии оценки результатов обучения по дисциплине
«удовлетворительно»/ «зачтено»	Выставляется обучающемуся, если он знает на базовом уровне теоретический и практический материал, допускает отдельные ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает определённые затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, владеет необходимыми для этого базовыми навыками и приёмами. Демонстрирует достаточный уровень знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне «достаточный».
«неудовлетворительно»/ «не зачтено»	Выставляется обучающемуся, если он не знает на базовом уровне теоретический и практический материал, допускает грубые ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает серьёзные затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, не владеет необходимыми для этого навыками и приёмами. Демонстрирует фрагментарные знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции на уровне «достаточный», закреплённые за дисциплиной, не сформированы.

6.3. Оценочные средства (материалы) для текущего контроля успеваемости, промежуточной аттестации обучающихся по дисциплине

Примерные варианты тестов:

ТЕСТ №1

Выберите один правильный вариант ответа из предложенных.

1. Что обязательно должна содержать любая исполняемая программа на C++?
A) Директиву `#include <iostream>`
B) Функцию `main()`
C) Библиотеку `math.h`
D) Оператор `return 0;`

2. Для чего используется директива `#include <iostream>`?
A) Для подключения математических функций
B) Для подключения средств ввода/вывода
C) Для объявления пространства имен
D) Для завершения программы

3. Какое пространство имен содержит стандартные функции C++?

A) `system`
B) `cpp`
C) `std`
D) `iostream`

4. Что означает строка `using namespace std;`?
A) Создание нового пространства имен
B) Подключение стандартной библиотеки
C) Объявление о том, что в коде будут использоваться имена из пространства имен `std`
D) Завершение программы

5. Какой оператор используется для проверки нескольких условий в C++?
A) `if-else`
B) `for`
C) `while`
D) `goto`

6. Что выведет следующий фрагмент кода?
```
int x = 5;
if (x = 10)
   cout << "Равно";
else
   cout << "Не равно";
```
A) Равно
B) Не равно
C) Ошибка компиляции
D) Ничего не выведет

7. Какое логическое выражение соответствует утверждению "x принадлежит интервалу (0, 10]"?
A) `x > 0 && x < 10`
B) `x > 0 && x <= 10`
C) `x >= 0 && x <= 10`
D) `x > 0 || x <= 10`

8. В каком случае используется оператор `switch`?
A) Для организации циклов
B) Для выбора одного из нескольких вариантов по значению выражения
C) Для объявления переменных
D) Для обработки исключений

9. Что произойдет, если в операторе `switch` не использовать `break` в блоке `case`?
A) Произойдет ошибка компиляции
B) Будет выполнен только текущий блок
C) Программа завершится аварийно
D) Выполнение "провалится" дальше в следующий блок

10. Какой цикл гарантированно выполнится хотя бы один раз?
A) `for`
B) `while`
C) `do-while`
D) Все циклы могут не выполниться ни разу

11. Сколько раз выполнится тело цикла?
```
for (int i = 0; i < 5; i++) {
   // тело цикла
}
```
A) 4 раза
B) 5 раз
C) 6 раз
D) Бесконечное количество раз

12. Что произойдет при выполнении этого кода?
```
int i = 1;
while (i > 0) {
   i++;
}
```
A) Цикл выполнится 1 раз
B) Цикл выполнится 10 раз
C) Произойдет зацикливание (бесконечный цикл)
D) Ошибка компиляции из-за отсутствия условия

13. Как можно прервать выполнение цикла досрочно?
A) Использовать `continue`
B) Использовать `break`
C) Использовать `return`
D) Использовать `exit`

14. Что делает оператор `continue`?
A) Завершает выполнение цикла
B) Завершает выполнение программы
C) Переходит к следующей итерации цикла, пропуская оставшийся код
D) Ничего не делает

15. Что такое прототип функции?
A) Тело функции
B) Объявление функции, содержащее ее имя, тип возвращаемого значения и параметры
C) Вызов функции
D) Переменная внутри функции

16. Каким способом передаются параметры в функцию по умолчанию в C++?
A) По ссылке
B) По указателю
C) По значению
D) По адресу

17. Что означает ключевое слово `void` перед именем функции?
A) Функция не принимает параметров
B) Функция ничего не возвращает
C) Функция возвращает целое число
D) Функция является шаблонной

18. Что такое рекурсивная функция?
A) Функция, которая вызывает саму себя

B) Функция, которая вызывает другую функцию
C) Функция без параметров
D) Функция, возвращающая вещественное число

19. Что хранит указатель?
A) Значение переменной
B) Имя переменной
C) Адрес переменной в памяти
D) Тип переменной

20. Как правильно объявить указатель на целое число?
A) `int ptr;`
B) `int &ptr;`
C) `int* ptr;`
D) `pointer int ptr;`

21. Для чего используется оператор `new`?
A) Для создания новой переменной на стеке
B) Для выделения памяти в куче (динамической памяти)
C) Для удаления переменной
D) Для получения адреса переменной

22. Какой индекс имеет первый элемент массива `int arr[10];`?
A) 1
B) 0
C) -1
D) Любой

23. Что произойдет при обращении к элементу массива за его границами?
A) Ошибка компиляции
B) Автоматическое расширение массива
C) Неопределенное поведение (может быть ошибка, может быть чтение "мусора")
D) Программа всегда вылетит с сообщением об ошибке

24. Как называется пользовательский тип данных, объединяющий несколько переменных разных типов?
A) Массив
B) Структура (`struct`)
C) Указатель
D) Перечисление (`enum`)

25. Какой класс используется для записи в файл в C++?
A) `ifstream`
B) `ofstream`
C) `fstream`
D) `file`

ТЕСТ №2
Инструкция: Выберите один правильный вариант ответа из предложенных.

1. Какой символ используется для завершения большинства инструкций в C++?
A) `.` (точка)

B) `,` (запятая)
C) `;` (точка с запятой)
D) `:` (двоеточие)

2. Как правильно закомментировать несколько строк кода?
A) `// комментарий //`
B) `<!-- комментарий -->`
C) `/* комментарий */`
D) `# комментарий`

3. Что означает `endl` в C++?
A) Конец файла
B) Пустую строку
C) Переход на новую строку и сброс буфера
D) Очистку экрана

4. Какая функция возвращает код возврата программы операционной системе?
A) `main()`
B) `return`
C) `exit`
D) `system`

5. Какое значение будет иметь переменная `y` после выполнения кода?
```
int x = 7;
int y = (x > 5) ? 10 : 20;
```
A) 5
B) 7
C) 10
D) 20

6. Как проверить, что переменная `ch` является заглавной латинской буквой?
A) `ch >= 'A' && ch <= 'Z'`
B) `ch > 'A' && ch < 'Z'`
C) `ch = 'A' || ch = 'Z'`
D) `ch >= A && ch <= Z`

7. Сколько раз выполнится проверка условия в этом коде?
```
int a = 5, b = 3;
if (a > b)
   if (a > 0)
      cout << "a положительное и больше b";
```
A) 1 раз
B) 2 раза
C) Ни разу
D) Зависит от значений

8. Что выведет код?
```
int a = 10;
if (a % 2 == 0)
   cout << "Четное";
else
   cout << "Нечетное";
```

A) Четное
B) Нечетное
C) Ошибка: % не работает с целыми числами
D) Ничего

9. Можно ли в операторе `switch` использовать переменную типа `double`?
A) Да, всегда
B) Да, но с ограничениями
C) Нет, только целочисленные типы и `char`
D) Нет, только `int`

10. Чем отличается цикл `while` от `do-while`?
A) `while` проверяет условие после выполнения тела, `do-while` — до
B) `while` проверяет условие до выполнения тела, `do-while` — после
C) Ничем, это синонимы
D) `while` используется только с числами, `do-while` — с символами

11. Что выведет этот код?
```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 2; j++) {
        cout << "*";
    }
    cout << endl;
}
```
A) `**\n**\n**\n`
B) `***\n***\n`
C) `******\n`
D) `*\n*\n*\n`

12. Как создать бесконечный цикл с помощью `for`?
A) `for(;;)`
B) `for(infinite)`
C) `for(while)`
D) `for(1)`

13. Что произойдет при выполнении?
```
int i = 0;
while(i < 5) {
    if(i == 3) continue;
    cout << i;
    i++;
}
```
A) Выведется `0124`
B) Выведется `012`
C) Произойдет зацикливание, так как `i++` не выполнится при `i == 3`
D) Ошибка компиляции

14. Сколько раз выполнится цикл?
```
int i = 10;
do {
    i--;
} while(i > 10);
```

A) 0 раз
B) 1 раз
C) 10 раз
D) Бесконечно

15. Что такое перегрузка функций?
A) Создание нескольких функций с разными именами
B) Создание нескольких функций с одинаковым именем, но разными параметрами
C) Вызов функции внутри другой функции
D) Слишком большое количество функций в программе

16. Как передать переменную в функцию так, чтобы функция могла изменить её значение?
A) Передать по значению
B) Передать по ссылке или указателю
C) Сделать переменную глобальной
D) Варианты B и C верны

17. Что возвращает функция, если в ней нет оператора `return`?
A) 0
B) -1
C) Неопределенное значение (мусор)
D) Ошибка компиляции

18. Для чего нужны параметры по умолчанию в функции?
A) Для автоматического определения типа параметра
B) Чтобы можно было вызывать функцию без указания всех аргументов
C) Чтобы увеличить скорость выполнения
D) Чтобы скрыть параметры от других функций

19. Как получить адрес переменной `x`?
A) `*x`
B) `&x`
C) `@x`
D) `addr(x)`

20. Что произойдет при выполнении `delete` для одного и того же указателя дважды?
A) Корректное освобождение памяти
B) Ничего не произойдет
C) Неопределенное поведение (обычно программа аварийно завершается)
D) Второй `delete` игнорируется

21. Чем ссылка отличается от указателя?
A) Ссылка не может быть перепривязана к другой переменной
B) Ссылка занимает больше памяти
C) Ссылка не требует инициализации
D) Указатель нельзя изменить

22. Как объявить двумерный массив 3x4?
A) `int arr[3,4];`
B) `int arr[3][4];`
C) `int arr[12];`
D) `int[3][4] arr;`

23. Имя массива в C++ — это:
A) Указатель на первый элемент массива
B) Переменная, хранящая размер массива
C) Специальный тип данных
D) Ссылка на последний элемент

24. Как обратиться к полю структуры, если есть указатель на структуру `ptr`?
A) `ptr.field`
B) `ptr->field`
C) `ptr::field`
D) `ptr.*field`

25. Какой режим открытия файла нужен для добавления данных в конец файла?
A) `ios::in`
B) `ios::out`
C) `ios::app`
D) `ios::trunc`

ТЕСТ №3
Инструкция: Выберите один правильный вариант ответа из предложенных.

1. Какая библиотека содержит математические функции (sin, cos, sqrt)?
A) `<iostream>`
B) `<cmath>` или `<math.h>`
C) `<string>`
D) `<algorithm>`

2. Что произойдет, если не написать `return 0;` в конце функции `main()`?
A) Ошибка компиляции
B) Программа не запустится
C) Компилятор автоматически добавит `return 0;` (в C++99 и новее)
D) Программа вернет случайное число

3. Какой тип данных используется для хранения одного символа?
A) `string`
B) `char`
C) `symbol`
D) `int`

4. Как объявить константу в C++?
A) `const int MAX = 100;`
B) `constant int MAX = 100;`
C) `#define MAX 100`
D) Варианты A и C верны (хотя A — более правильный для C++)

5. Что такое "висячий else"?
A) Ситуация, когда else относится к ближайшему if
B) else без if
C) Ошибка компиляции
D) Специальный оператор

6. Какое условие нужно записать, чтобы проверить, что год `y` является високосным? (Високосный: делится на 4, но не на 100, кроме случаев, когда делится на 400)
A) `y % 4 == 0`
B) `(y % 4 == 0 && y % 100 != 0) || y % 400 == 0`
C) `y % 4 == 0 && y % 100 == 0 && y % 400 == 0`
D) `y % 400 == 0`

7. Что выведет код?
```
int a = 5;
int b = 10;
if (a > b)
   if (a > 0) cout << "A";
else cout << "B";
```
A) A
B) B
C) Ничего
D) Ошибка

8. Какой оператор проверяет равенство двух значений?
A) `=`
B) `= =`
C) `! =`
D) `= = =`

9. Можно ли в `switch` использовать несколько меток `case` для одного блока кода?
A) Нет, каждая метка должна иметь свой код
B) Да, можно написать `case 1: case 2: cout << "1 или 2"; break;`
C) Только если использовать `goto`
D) Да, но только для символьных констант

10. Какой цикл лучше всего подходит для перебора элементов массива, когда известно их количество?
A) `while`
B) `do-while`
C) `for`
D) Любой

11. Что выведет код?
```
int i = 0;
while(i < 5) {
   i++;
   if(i == 3) break;
   cout << i << " ";
}
```
A) `1 2 3`
B) `1 2`
C) `0 1 2`
D) `1 2 3 4 5`

12. Сколько раз выполнится цикл?
```
for(int i = 10; i > 0; i--);
```
A) 9 раз

B) 10 раз
C) 11 раз
D) Ни разу, так как после for сразу стоит точка с запятой

13. В какой части цикла `for` можно объявить переменную?
A) Только в инициализации
B) Только в условии
C) Только в итерации
D) В любой части

14. Что произойдет при выполнении этого кода?
```
int i = 0;
while(i < 5);
   i++;
```
A) Цикл выполнится 5 раз
B) Цикл выполнится 1 раз
C) Бесконечный цикл
D) Ошибка компиляции

15. Что такое сигнатура функции?
A) Тело функции
B) Имя функции и список параметров
C) Тип возвращаемого значения
D) Адрес функции в памяти

16. Может ли функция возвращать массив?
A) Да, всегда
B) Нет
C) Может вернуть указатель на массив
D) Только если массив глобальный

17. Что такое inline-функция?
A) Функция, определенная внутри другой функции
B) Функция, код которой подставляется непосредственно в место вызова для ускорения
C) Функция без имени
D) Функция, работающая с потоками ввода/вывода

18. Что будет, если вызвать рекурсивную функцию без базового случая (условия выхода)?
A) Ничего особенного
B) Функция выполнится один раз
C) Произойдет переполнение стека (stack overflow)
D) Компилятор исправит ошибку

19. Что такое "утечка памяти"?
A) Потеря данных при выключении компьютера
B) Ситуация, когда программа не освобождает выделенную память
C) Слишком быстрое заполнение памяти
D) Вирусная атака

20. Как правильно освободить память, выделенную под массив `int* arr = new int[10];`?
A) `delete arr;`
B) `free(arr);`

C) `delete[] arr;`
D) `delete arr[];`

21. Что выведет код?
int x = 5;
int& ref = x;
ref = 10;
cout << x;
A) 5
B) 10
C) Адрес x
D) Ошибка

22. Как передать двумерный массив в функцию?
A) `void func(int arr[][10], int rows)`
B) `void func(int** arr, int rows, int cols)`
C) `void func(int arr[10][10])`
D) Все варианты возможны в разных контекстах

23. Как вычислить количество элементов в статическом массиве `int arr[20];`?
A) `sizeof(arr)`
B) `sizeof(arr) / sizeof(arr[0])`
C) `length(arr)`
D) `arr.size()`

24. Чем структура (`struct`) отличается от объединения (`union`) в C++?
A) В `union` все поля хранятся одновременно, в `struct` — по очереди
B) В `union` все поля используют одну область памяти, в `struct` — каждый свою
C) Ничем, это синонимы
D) `union` нельзя использовать в функциях

25. Как проверить, успешно ли открылся файл?
A) `if (file)`
B) `if (file.is_open())`
C) `if (file.good())`
D) Все варианты верны

Критерии оценки: 90-100% — "отлично", 75-89% — "хорошо", 60-74% — "удовлетворительно", менее 60% — "неудовлетворительно".

ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К ЗАЧЕТУ:

1. Состав языка C++. Алфавит языка. Служебные (ключевые) слова. Знаки операций. Разделители. Идентификаторы.
2. Структура программы на языке C++. Директивы препроцессора. Глобальные объявления. Функция `main()`. Блоки операторов.
3. Переменные и идентификаторы. Правила создания идентификаторов. Понятие переменной. Объявление переменных. Область видимости переменных.
4. Типы данных языка C++. Целочисленные типы (`int`, `short`, `long`, `long long`). Символьный тип (`char`). Вещественные типы (`float`, `double`). Логический тип (`bool`). Размеры типов (оператор `sizeof`).

5. Описание констант и переменных. Инициализация переменных. Литералы (целые, вещественные, символьные, строковые). Константы (`const`). Способы инициализации переменных (присваивание, прямая инициализация, универсальная инициализация `{}`).
6. Основные операции языка C++. Арифметические операции. Операции сравнения. Логические операции. Операции присваивания. Приоритет операций.
7. Директива препроцессора `#include`. Назначение. Формы записи (с угловыми скобками и кавычками). Поиск подключаемых файлов.
8. Ввод-вывод с использованием библиотеки `iostream`, `iomanip`. Потоки `cin`, `cout`, `cerr`, `clog`. Операторы `<<` и `>>`. Форматирование вывода: манипуляторы `endl`, `setw()`, `setprecision()`, `fixed`, `left`, `right`.
9. Директива препроцессора `#define`. Определение макроконстант. Макроопределения с параметрами (опасность использования, отличие от `const` и функций).
10. Библиотека математических функций `math.h` / `cmath`. Основные функции: `sqrt()`, `pow()`, `sin()`, `cos()`, `tan()`, `exp()`, `log()`, `fabs()`. Подключение библиотеки.
11. Условный оператор `if`. Полная и сокращенная форма. Вложенные условные операторы. Проблема "висячего `else`". Использование составных операторов (блоков `{}`).
12. Множественный выбор: оператор `switch`. Синтаксис. Структура `case`-меток. Использование `break` для предотвращения "проваливания". Ключевое слово `default`.
13. Инструкции перехода (`goto`, `continue`, `return`, `break`). Назначение и особенности применения. `break` в циклах и `switch`. `continue` в циклах. `return` для выхода из функции. Критика использования `goto`.
14. Оператор цикла с предусловием (`while`). Синтаксис. Особенности выполнения (возможно, ни одного выполнения). Типичные ошибки (отсутствие изменения условия).
15. Оператор цикла с постусловием (`do-while`). Синтаксис. Отличие от `while` (гарантированное выполнение хотя бы один раз). Области применения.
16. Решение задач циклической структуры. Табулирование функций (вывод значений функции на интервале с заданным шагом). Вычисление сумм и произведений числовых рядов (конечных и бесконечных с заданной точностью).
17. Оператор цикла с заданным числом повторений (`for`). Синтаксис. Инициализация, условие, итерация. Множественные выражения в частях цикла. Вложенные циклы.
18. Одномерные массивы. Объявление. Инициализация (списком, частичная, автоматическая). Доступ к элементам по индексу. Ввод и вывод значений элементов.
19. Использование датчика случайных чисел. Функции `srand()` и `rand()`. Генерация псевдослучайных чисел в заданном диапазоне. Заполнение массивов случайными значениями.
20. Решение задач обработки числовых значений одномерных массивов.
Нахождение суммы и произведения всех элементов или элементов, удовлетворяющих условию.
Нахождение максимального (минимального) значения и его индекса.
Подсчет количества элементов по заданному критерию.
21. Решение задач переупорядочивания элементов в массиве (сортировка).
Метод "пузырька" (bubble sort).
Метод прямого выбора (selection sort).
Понятие об устойчивости сортировки и вычислительной сложности (O-большое).
22. Двумерные массивы. Объявление и инициализация. Интерпретация как матрицы. Ввод значений (по строкам). Вывод в общепринятом табличном виде.
23. Решение задач обработки двумерных массивов. Обработка по строкам. Обработка по столбцам. Поиск элементов на главной и побочной диагоналях. Транспонирование матриц.
24. Строки в стиле C (массивы символов). Объявление и инициализация. Строковые литералы. Функции библиотеки `cstring` / `string.h`: `strlen()`, `strcpy()`, `strcat()`, `strcmp()`.
25. Решение задач обработки символьных строк. Подсчет символов. Поиск подстроки. Замена символов. Проверка на палиндром.

26. Определение и вызов пользовательских функций. Синтаксис определения. Тип возвращаемого значения. Тело функции. Механизм вызова и возврата управления.
27. Понятие прототипа функции. Назначение (объявление функции до ее определения). Синтаксис прототипа. Расположение прототипов (в заголовочных файлах или в начале программы).
28. Формальные и фактические параметры функции. Формальные параметры в определении. Фактические параметры при вызове. Соответствие по типу и количеству.
29. Понятие рекурсии. Рекурсивные алгоритмы. Прямая и косвенная рекурсия. Базовый случай (условие выхода). Рекурсивный шаг. Примеры: факториал, числа Фибоначчи. Достоинства и недостатки (наглядность vs. расход памяти).
30. Объявление, инициализация указателей, операции с указателями. Операторы `*` (разыменование) и `&` (взятие адреса). Нулевой указатель (`nullptr`). Арифметика указателей.
31. Связь массивов и указателей. Имя массива как константный указатель на первый элемент. Доступ к элементам через указатель (индексация и адресная арифметика).
32. Передача массива в функцию. Особенности передачи (всегда по адресу). Необходимость передачи размера массива отдельным параметром.
33. Решение задач обработки массивов через указатели. Итерация по массиву с помощью указателя. Использование указателей в функциях для модификации массивов.
34. Ссылки. Передача аргументов функции по ссылке. Объявление ссылки. Отличие ссылки от указателя. Преимущества передачи по ссылке (избежание копирования, возможность изменения). Константные ссылки.
35. Методы распределения динамической памяти. Динамические массивы. Статическая, автоматическая и динамическая память. Операторы `new` и `delete` для одиночных объектов. Операторы `new[]` и `delete[]` для массивов. Создание динамических массивов (одномерных и двумерных). Утечки памяти.
36. Перечисляемый тип (`enum` и `enum class`). Объявление. Значения элементов. Преимущества `enum class` (строгая типизация).
37. Объявления `typedef`. Создание псевдонимов для существующих типов. Использование `using` (C++11) как альтернативы `typedef`.
38. Объявление структурного шаблона и структурной переменной. Инициализация. Ключевое слово `struct`. Поля структуры. Создание переменных структурного типа. Инициализация (агрегатная, через конструктор по умолчанию).
39. Работа со структурами через указатели. Оператор `->` для доступа к полям. Передача структур в функцию по указателю.
40. Объединения (`union`). Особенности распределения памяти (все поля разделяют одну область памяти). Размер объединения. Практическое применение.
41. Решение задач обработки структурированных данных. Создание массива структур. Поиск, сортировка, добавление и удаление записей в массиве структур.
42. Текстовые и бинарные файлы. Понятие файла. Отличия текстового и бинарного форматов хранения. Преобразования при записи/чтении (в текстовом режиме).
43. Функции работы с текстовыми файлами. Классы `ifstream`, `ofstream`, `fstream`. Методы `open()`, `close()`, `is_open()`. Чтение по словам, по строкам (`getline()`), посимвольно (`get()`). Запись (`<<`).
44. Функции работы с бинарными файлами. Режимы открытия: `ios::binary`. Методы `read()` и `write()`. Прямой доступ: `seekg()`, `seekp()`, `tellg()`, `tellp()`.
45. Решение задач обработки данных, организованных в файлы. Сохранение массива структур в файл (текстовый и бинарный). Загрузка данных из файла. Поиск записи в файле. Модификация данных в файле.

ПРИМЕРНЫЕ ТЕМЫ ДОПОЛНИТЕЛЬНЫХ ПИСЬМЕННЫХ РАБОТ:

1. Линейные алгоритмы и ввод-вывод данных
2. Разветвляющиеся алгоритмы
3. Циклические алгоритмы
4. Одномерные массивы: ввод, вывод и базовые алгоритмы
5. Сортировка и модификация массивов
6. Двумерные массивы (матрицы)
7. Обработка символьных строк (С-строки)
8. Пользовательские функции и параметры
9. Рекурсивные алгоритмы
10. Указатели и адресная арифметика
11. Динамические массивы и управление памятью
12. Структуры и массивы структур
13. Перечисления и объединения
14. Работа с текстовыми файлами
15. Работа с бинарными файлами и прямым доступом
16. Комплексная обработка структурированных данных (мини-проект)
17. Алгоритмы на графах и матрицах (для продвинутых групп)
18. Численные методы (для прикладной направленности)

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

Тема: Структура программы на языке С++. Ввод и вывод данных

Цель занятия: изучить структуру программы на С++, освоить базовые операции ввода/вывода, научиться работать с основными типами данных и форматированием вывода.

Вопросы для обсуждения

1. Структура программы на С++. Из каких обязательных элементов состоит программа? Что такое директива препроцессора? Для чего нужна функция `main()` и какое значение она возвращает?
2. Пространства имен. Что такое пространство имен `std`? Зачем нужно объявление `using namespace std;`? Какие альтернативы существуют?
3. Базовые типы данных. Какие типы данных существуют в С++? Каковы их размеры в памяти и диапазоны допустимых значений? От чего зависит размер типа?
4. Ввод и вывод данных. Как работают потоки `cin` и `cout`? В чем разница между `cout << endl;` и `cout << "\n";`?
5. Форматирование вывода. Какие манипуляторы из библиотеки `<iomanip>` позволяют управлять шириной поля, точностью вывода вещественных чисел и выравниванием?
6. Обработка ошибок ввода. Что произойдет, если пользователь введет букву вместо числа? Как проверить корректность ввода и очистить поток при ошибке?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2

Тема: Разветвляющиеся программы. Условные операторы

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

Тема: Операторы цикла

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4

Тема: Одномерные массивы и алгоритмы их обработки

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5

Тема: Двумерные массивы (матрицы)

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6

Тема: Пользовательские функции

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7
Тема: Рекурсивные функции

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8
Тема: Указатели и ссылки

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9
Тема: Динамическое выделение памяти

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №10
Тема: Типы данных, определяемые пользователем. Структуры