Фонд оценочных средств для промежуточной аттестации по дисциплине

Программирование

Направление подготовки (специальность)

09.03.04 Программная инженерия

Направленность (профиль)

Разработка программно-информационных систем

Присваиваемая квалификация (степень)

бакалавр

Форма обучения

заочная форма обучения

Год(ы) набора 2026

Челябинск 2026 г.

09.03.04 Программная инженерия профиль Разработка программно-информационных систем, дисциплина Программирование, 2026 год набора, заочная форма обучения

Фонд оценочных средств дисциплины (модуля) одобрен и рекомендован:
Проректор по учебной работе утверждено 27.02.2026 А.А. Саламатов

Ученым советом института информационных технологий

Протокол заседания № 7 от 26.02.2026

Председатель Ученого совета
института информационных
технологий согласовано Ю.В. Петриченко

Заседанием кафедры информационных технологий и экономической информатики

Протокол заседания №7 от 26.02.2026

Заведующий кафедрой согласовано С.А. Скрипов

Автор (составитель) И.Е. Николаев

Структура фондов оценочных средств соответствует приказу ректора ФГБОУ ВО «ЧелГУ» от 27 сентября 2022 № 573-1

Содержание

1. Паспорт фонда оценочных средств

Направление подготовки: 09.03.04 Программная инженерия
Направленность: Разработка программно-информационных систем
Дисциплина: Программирование
Курс (курсы) изучения: 1, 2
Форма промежуточной аттестации: экзамены
Для оценивания результатов обучения используется балльно-рейтинговая система.

2. Перечень формируемых компетенций

Изучение дисциплины «Программирование» направлено на формирование компетенций, приведённых в 1.

Таблица 1. Результаты обучения по дисциплине.

Коды компетенции согласно ФГОС (ОПОП ВО)	Содержание компетенций согласно ФГОС (ОПОП ВО)	Индикаторы достижения компетенции согласно ОПОП	Перечень планируемых результатов обучения по дисциплине
1	2	3	4
ОПК-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности;	ОПК-1.1. Демонстрирует знание основных положений и концепций в области математических и естественных наук, вычислительной техники и программирования ОПК-1.2. Демонстрирует умения решать стандартные задачи в профессиональной деятельности с применением естественнонаучных и общеинженерных знаний, методов математического анализа и моделирования ОПК-1.3. Имеет практический опыт применения основных теорем и законов математики и. естественных наук, методов моделирования, теоретического и экспериментального исследования для решения задач профессиональной деятельности	Знать:- об основных средах разработки программного обеспечения и их особенностях- знать основные технологии работы с базами данных, сетевых технологий. Уметь:- работать в среде программирования (составление, отладка и тестирование программ). Владеть:- навыками работы в различных средах программирования.
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;	ОПК-6.1. Демонстрирует знание основ информатики, теории алгоритмов, методологии и технологии программирования ОПК-6.2. Демонстрирует умения разрабатывать алгоритмические и программные решения, проводить проектирование, конструирование и тестирование программных продуктов ОПК-6.3. Имеет практический опыт использования технологий разработки программного обеспечения	Знать:- знать методы проектирования и разработки модульных программ- знать основные технологии разработки интерфейсов программ Уметь:- применять методы проектирования и разработки с использование различных методологии программирования- разрабатывать многомодульные программ Владеть:- навыками разработки программ сложной архитектуры- навыками отладки и тестирования программ
ПК-1	Владение навыками использования операционных систем, сетевых технологий, современных языков программирования,	ПК-1.1. Демонстрирует знание основ операционных систем, сетевых технологий, языков программирования, баз данных и технологий обработки данных, основ проектирования интерфейсов, языков и методов формальных спецификаций ПК-1.2. Демонстрирует умения разрабатывать	Знать:- особенности конструировании алгоритмов- абстракции основных структур данных (списки, множества и т.п.) и методы их обработки и способах реализации- основные понятия и концепции

	технологий обработки данных, средств разработки программного интерфейса, применения языков и методов формальных спецификаций, систем управления базами данных	системное и прикладное программного обеспечение с использованием языков и технологий программирования, баз данных, сетевых технологий и операционных систем, языков и методов формальных спецификаций ПК-1.3. Имеет практический опыт использования операционных систем, современных языков программирования, систем управления базами данных и технологий обработки данных, средств разработки программного интерфейса	структурной и объектно-ориентированной парадигмы Уметь:- уметь применять знания из других профессиональных областей при разработке собственных программ - уметь применять знания инструментов моделирования на практике для работы программ Владеть:- навыками применения знаний естественно научных и инженерных дисциплин в своих программах

3. Содержание оценочных средств по дисциплине

3.1.Виды оценочных средств

Таблица 2. Виды оценочных средств.

№ п/п	Код компетенции/ планируемые результаты обучения	Контролируемые темы/ разделы	Наименование оценочного средства для текущего контроля	Наименование оценочного средства на промежуточной аттестации/№ задания
1	ОПК-1.1. Демонстрирует знание основных положений и концепций в области математических и естественных наук, вычислительной техники и программирования Знать:- об основных средах разработки программного обеспечения и их особенностях- знать основные технологии работы с базами данных, сетевых технологий.	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
2	ОПК-1.2. Демонстрирует умения решать стандартные задачи в профессиональной деятельности с применением естественнонаучных и общеинженерных знаний, методов математического анализа и моделирования Уметь:- работать в среде программирования (составление, отладка и тестирование программ).	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
3	ОПК-1.3. Имеет практический опыт применения основных теорем и законов математики и. естественных наук, методов моделирования, теоретического и экспериментального исследования для решения задач профессиональной деятельности Владеть:- навыками работы в различных средах программирования.	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
4	ОПК-6.1. Демонстрирует знание основ информатики, теории алгоритмов, методологии и технологии программирования Знать:- знать методы проектирования и разработки модульных программ- знать основные технологии разработки интерфейсов программ	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень)	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160

		Классификация ВС - М. Флинна		
5	ОПК-6.2. Демонстрирует умения разрабатывать алгоритмические и программные решения, проводить проектирование, конструирование и тестирование программных продуктов Уметь:- применять методы проектирования и разработки с использование различных методологии программирования- разрабатывать многомодульные программ	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
6	ОПК-6.3. Имеет практический опыт использования технологий разработки программного обеспечения Владеть:- навыками разработки программ сложной архитектуры- навыками отладки и тестирования программ	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
7	ПК-1.1. Демонстрирует знание основ операционных систем, сетевых технологий, языков программирования, баз данных и технологий обработки данных, основ проектирования интерфейсов, языков и методов формальных спецификаций Знать:- особенности конструировании алгоритмов- абстракции основных структур данных (списки, множества и т.п.) и методы их обработки и способах реализации- основные понятия и концепции структурной и объектно-ориентированной парадигмы	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
8	ПК-1.2. Демонстрирует умения разрабатывать системное и прикладное программного обеспечение с использованием языков и технологий программирования, баз данных, сетевых технологий и операционных систем, языков и методов формальных спецификаций Уметь:- уметь применять знания из других профессиональных областей при разработке собственных программ - уметь применять знания инструментов моделирования на практике для работы программ	Блок-схема алгоритма. Введение в язык программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна	Тест	Семестр 2. Задания теста № 1-167 Семестр 3. Задания теста № 1-160
9	ПК-1.3. Имеет практический опыт использования операционных систем,	Блок-схема алгоритма. Введение в язык	Тест	Семестр 2. Задания теста № 1-167

	современных языков программирования, систем управления базами данных и технологий обработки данных, средств разработки программного интерфейса Владеть:- навыками применения знаний естественно научных и инженерных дисциплин в своих программах	программирования Архитектура Фон-Неймана. Оптимизация подсистемы памяти. Кэш-память Язык программирования C# (продвинутый уровень) Классификация ВС - М. Флинна		Семестр 3. Задания теста № 1-160

Типовые задания, критерии и показатели оценивания в рамках текущего контроля представлены в рабочей программе дисциплины (модуля).

3.2.Содержание оценочных средств

База тестовых вопросов для 2 семестра

№ п/п	Формулировка вопроса	Варианты ответов (полужирным шрифтом – верные варианты)
1.	typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;int main(){ A *a1 = (A*)malloc(sizeof(A)); A *a2 = (A*)malloc(sizeof(A)); A *a3 = (A*)malloc(sizeof(A)); a1->link=a2; a2->link=a3;}как правильно получить адрес нулевого элемента?	**a. a1** b. *a1 c. &a1 d. a1[0]
2.	Дана структура:struct Node{ int data; Node *next;}Структура типа Node может использоваться для описания элементов	**a. односвязного списка** b. двухсвязного списка c. дерева
3.	Дана структура:struct Node{ int data; Node *left; Node *right;}Структура типа Node может использоваться для описания элементов	a. односвязного списка **b. двухсвязного списка**
4.	Каждый элемент линейного односвязного списка представляется структурой:typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;void ????(UZEL *p) { UZEL *x; x = p -> link; //получ узел if (x != NULL) { p -> link = x -> link; //перенаправляем связи free (x); } return;}Эта функция -	**a. удаляет элемент из списка** b. вставляет элемент из списка c. удаляет весь список d. заполняет список нулями
5.	Каждый элемент линейного односвязного списка представляется структурой:typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;UZEL* ?????(UZEL *p) { UZEL *x; //объявляем новый узел x = (UZEL*) calloc (1, sizeof(UZEL)); //распределяем память if (x = = NULL) авария(); //если память не выделилась x -> info = новые данные; //заполнение данными узла x -> link = p link; //перенаправляем связи p -> link = x; return x;}Эта функция	a. удаляет элемент из списка **b. вставляет элемент из списка** c. удаляет весь список d. заполняет список нулями
6.	typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;int main(){ A a1; A a2; A a3; a1.link=&a2; a2.link=&a3;}Является ли конструкция списком?	**a. да** b. нет
7.	typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;int main(){ A a1;	a. да **b. нет**

	A a2; A a3;}Является ли конструкция списком?	
8.	typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;int main(){ A *a1 = (A*)malloc(sizeof(A)); A *a2 = (A*)malloc(sizeof(A)); A *a3 = (A*)malloc(sizeof(A)); a1->link=a2; a2->link=a3;}Является ли конструкция списком?	**a. Да** b. Нет
9.	typedef struct A { ... info; //поле данных struct A *link; //поле связи, указатель на следующий элемент} UZEL;int main(){ A *a1 = (A*)malloc(sizeof(A)); A *a2 = (A*)malloc(sizeof(A)); A *a3 = (A*)malloc(sizeof(A));}Является ли конструкция списком?	a. Да **b. Нет**
10.	с помощью какой функции stdio.h можно открыть текстовый файл для чтения?	**a. fopen** b. fileopen c. openf d. fopenf e. fopenread
11.	какой строчкой мы откроем текстовый файл для чтения	**a. f=fopen("text.txt","rt")** b. f=fopen("text.txt","wt") c. f=fopen("text.txt","rt++") d. f=fopen("text.txt","rb") e. f=fopen("text.txt","tr")
12.	какой строчкой мы откроем текстовый файл для записи	a. f=fopen("text.txt","rt") **b. f=fopen("text.txt","wt")** c. f=fopen("text.txt","rt++") d. f=fopen("text.txt","wb") e. f=fopen("text.txt","tw")
13.	Каким атрибутом передается режим открытия файла, например "rt", в функции fopen?	a. первым **b. вторым** c. третьим d. можно передать хоть 1-ым, хоть 2-ый, хоть 3-им
14.	Каким атрибутом передается имя файла в функции fopen?	**a. первым** b. вторым c. третьим d. можно передать хоть 1-ым, хоть 2-ый, хоть 3-им
15.	что возвращает функция fopen в качестве результата?	a. сам файл b. все данный внутри файла **c. указтаель на блок управления файлом** d. указатель на файл e. целочисленную переменную
16.	Что за переменная a?int main(){ . . . int a; . . .}	**a. Автоматическая переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная
17.	Что за переменная a?int main(){ . . . int a; . . .}	**a. Локальная переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная

18.	Что за переменная a?int main(){ . . . static int a; . . .}	**a. Статическая переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная
19.	Что за переменная a?int a;int main(){ . . .}	**a. Глобальная переменная** b. Локальная переменная c. Динамическая переменная d. Константа
20.	Какая память является стековой?	**a. Автоматическая** b. Глобальная-статическая c. Динамическая
21.	какая переменная будет находиться на дне стека в момент исполнения программы в точке ТУТ?int a; int main(){int b;int c;{int d;+++ ТУТ +++} }	a. a **b. b** c. c d. d
22.	какая переменная будет находиться на вершине стека?

int a;int main(){int b;int c; {int d; +++ ТУТ +++ } } | a. a
b. b
c. c
d. d |
| 23. |

какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ?int a; int main(){ int b; int c; { int d; } --- ТУТpintf …} | a. a
b. b
c. c
d. d |
| 24. | какая переменная будет находиться в самом старшем адресе?int a; int main(){int b;int c;{int d; +++ ТУТ +++} } | **a. a**
b. b
c. c
d. d |
| 25. | какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ

 Что будет выведено на экран в результате работы программы #include <stdio.h>#include <stdlib.h> int f(){ static int a=0; a=a+1; printf("%i",a);} int main(){ f(); f(); f(); system("pause"); return 0;} | **a. 123**
b. 12
c. 111
d. 11
e. 000 |
| 26. | какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ

 Что будет выведено на экран в результате работы программы #include <stdio.h>#include <stdlib.h> int f(){ int a=0; a=a+1; printf("%i",a);} int main(){ f(); f(); f(); system("pause"); return 0;} | a. 123
b. 12
c. 111
d. 11
e. 000 |
| 27. | В какой момент произойдет выделение памяти под глобальную переменную? | **a. В момент запуска программы**
b. при входе в функцию, где переменная используется
c. При первом обращении к переменной |
| 28. | В какой момент произойдет выделение памяти под переменную static? | **a. В момент запуска программы**
b. при входе в функцию, где переменная используется
c. При первом обращении к переменной
d. При инициализации переменной |

29.	В какой момент произойдет выделение памяти под автоматическую переменную?	a. В момент запуска программы b. при входе в функцию, где переменная используется c. При первом обращении к переменной d. При инициализации переменной **e. В точке объявления переменной**
30.	В какой момент произойдет удаление памяти из под глобальной переменной?	**a. В момент завершения программы** b. После момента, когда переменная больше не используется c. При выходе из функции, где эта переменная используется d. Не удаляется никогда, занимая место в оперативной памяти
31.	В какой момент произойдет удаление памяти из под автоматической переменной?	a. В момент завершения программы b. только привыходе из функции где она объявлена **c. При выходе из блока, где переменная объявлена** d. При выходе из любой функции, неважно где переменная объявлена
32.	Какой способ хранения переменных самый эффективный с точки зрения скорости распределения памяти?	**a. Статический - в глобальной памяти** b. автоматический - в стеке c. динамический - в куче
33.	Какая функция, команда не используется для динамического выделения памяти?	**a. mem** b. malloc c. calloc d. new
34.	Какая функция, команда не используется для высвобождения динамически выделенной памяти?	**a. destroy** b. delete c. free
35.	Каких функций нет в стандартном СИ (не С++) для работы с динамической памятью?	**a. new** **b. delete** c. malloc d. calloc e. free
36.	Что делает эта строчка программы: int *p = (int*) malloc(400); ?	a. ничего не делает **b. Выделяет 400 байт и записывает адресс выделенного блока в переменную p** c. Выделяет массив из 400 элементов типа int и записывает адрес выделенного блока в переменную p d. Выделяет 400 бит и записывает адрес выделенного блока в переменную p e. free
37.	из какой области памяти происходит выделение памяти для динамических переменных?	**a. из кучи** b. из области памяти стека c. из глоабльной области памяти

38.	В какой библиотеке описаны функции free и malloc?	**a. stdlib.h** b. conio.h c. windows.h d. DSL.h
39.	Сколько байт выделится с следующей строчке:int* p = (int*) malloc(100*sizeof(int));	**a. 400** b. 100 c. 300 d. 800 e. 50
40.	Сколько байт выделится с следующей строчке:char* p = (char*) malloc(100*sizeof(char));	a. 400 **b. 100** c. 300 d. 800 e. 50
41.	Какое значение вернет функция malloc или new если не сможет выделить память?	**a. NULL** b. error c. -1 d. 15
42.	Что такое NULL?	**a. Константа со значением 0** b. Переменная со значением 0 c. Константа со значением -1 d. Текстовая строка
43.	что означает NULL?	**a. пустой указатель** b. указатель на стек c. константа, показывающая ошибку при очистке памяти
44.	int *p = (int*) malloc(400);Можно ли проверить по значению указателя, произошла ли ошибка при выделении памяти?	**a. ДА** b. НЕТ
45.	int main(){ int *p = (int*) malloc(400); *(p+1)=4; free(p);}есть ли тут ошибка?	**a. НЕТ** b. ДА
46.	В отличие от поименованных переменных динамические переменные ...	**a. Не удаляются при выходже из блока** b. НЕТ ОЛИЧИЙ ОТ ПОИМЕНОВАННЫХ ПЕРЕМЕННЫХ c. хранятся в стеково памяти d. хранятся внутри кода программы в памяти
47.	int main(){ int *p = (int*) malloc(400); *(p+1)=4; free(p+1);}есть ли тут ошибка?	**a. ДА** b. НЕТ
48.	Какие из перечисленных АТД являются линейными:	**a. Список** **b. Очередь** c. Дерево d. Сеть
49.	Можно ли составить сортированный список обычных массивов?	a. Да **b. Нет**

50.	Нужно ли в структуре "Связный список" хранить количество элементов?	a. Да **b. Нет**
51.	Обязывает ли АТД "Список" хранить количество добавленных элементов внутри реализующей его структуры?	a. Да **b. Нет**
52.	Можно ли представить реализацию АТД "Очередь" без сохранения информации о количестве элементов в ней?	**a. Да** b. Нет
53.	Где быстрее работает получение элемента по индексу?	a. В связном списке **b. В списке на основе массива** c. В двусвязном списке
54.	Какие структуры не являются динамическими:	**a. Запись** **b. Автоматический массив (int M[100])** c. Связный список d. B-Дерево
55.	Вам нужно найти пересечение двух введённых пользователем последовательностей чисел.Какой из перечисленных АТД Вы выберете?	a. Стек b. Очередь c. Карта **d. Множество** e. Список
56.	Вам нужно где-то сохранить введённую пользователем последовательность чисел.Какой из перечисленных АТД Вы выберете?	a. Стек b. Очередь c. Карта d. Множество **e. Список**
57.	Вам нужно вывести введённую пользователем последовательность чисел в обратном порядке.Какой из перечисленных АТД Вы выберете?	**a. Стек** b. Очередь c. Карта d. Множество e. Список
58.	Вам нужно организовать конвеерную обработку поступающих от пользователя запросов.Какой из перечисленных АТД Вы выберете?	a. Стек **b. Очередь** c. Карта d. Множество e. Список
59.	Можно ли составить очередь из списков?	**a. Да** b. Нет
60.	Можно ли составить стек массивов?	**a. Да** b. Нет
61.	Переведите в дополнительный однобайтный код-128 = (?)доп код	**a. 10000000** b. 11111111 c. 11000000 d. 01111111 e. 100000000
62.	Переведите в дополнительный однобайтный код-1 = (?)доп код	a. 10000001

		b. 11111110 c. 11000001 d. 01111111 **e. 111111110** **f. 11111111**
63.	Переведите в дополнительный однобайтный код-1 = (?)доп код	a. 10000001 b. 11111110 c. 11000001 d. 01111111 **e. 11111111**
64.	Что будет выведено на экран?char a=255;printf("%i",a);	**a. -1** b. 1 c. 255 d. ошибка
65.	Что будет выведено на экран?char a=(10000001); - задано в двоичном кодеа=a-1;printf("%i",a);	**a. -128** b. 128 c. -127 d. 127 e. 10000000 f. -64 g. 64 h. -1
66.	Что будет выведено на экран?unsigned char a=(10000001); - задано в двоичном кодеа=a-1;printf("%i",a);	**a. 128** b. -128 c. -127 d. 127 e. 10000000 f. -64 g. 64 h. -1
67.	Что будет выведено на экран?unsigned char a=0;a=a-2;printf("%i",a);	**a. 254** b. -2 c. 255 d. ошибка
68.	Что будет выведено на экран?unsigned char a=255;a=a-1;printf("%i",a);	**a. 254** b. -2 c. 255 d. ошибка
69.	Что будет выведено на экран?unsigned char a=-2;printf("%i",a);	**a. 254** b. 1 c. -2 d. ошибка e. 255 f. 11111110
70.	Что будет выведено на экран?int a=-129;printf("%i",a);	**a. -129** b. 127 c. 254 d. 64

		e. 1
71.	Что будет выведено на экран?char a=-129;printf("%i",a);	**a. 127** b. 256 c. 254 d. 64 e. 1
72.	Что будет выведено на экран?char a=-3 & 3;printf("%i",a);	a. -3 b. -1 c. -0.3 d. 3 **e. 1** f. 2
73.	Что будет выведено на экран?char a=-3 & 3;printf("%i",a);	a. -3 b. -1 c. -0.3 d. 3 **e. 1** f. 2
74.	Что будет выведено на экран?unsigned char a=-2 | -1;printf("%i",a);	**a. 255** b. -2 c. -1 d. 2 e. 1 f. 3 g. -3
75.	Что будет выведено на экран?char a=-3 |-1;printf("%i",a);	a. -3 **b. -1** c. -0.3 d. 3 e. 1
76.	Что будет выведено на экран?char a=-3 & -1;printf("%i",a);	**a. -3** b. -1 c. -0.3 d. 3 e. 1
77.	Что будет выведено на экран?char a=-4;a=a>>3;printf("%i",a);	**a. -1** b. 1 c. -0.5 d. 0.5 e. 31
78.	Что будет выведено на экран?unsigned char a=255;a=a>>1;printf("%i",a);	**a. 127** b. 256 c. 254 d. 64 e. 1
79.	Что будет выведено на экран?char a=64;a=a<<1;printf("%i",a);	a. 4 b. 1

		c. -0.5 d. 0.5 e. 31 f. 128 **g. -128**
80.	Что будет выведено на экран?char a=2;a=a<<1;printf("%i",a);	**a. 4** b. 1 c. -0.5 d. 0.5 e. 31
81.	Сложить в 11-ой CCA5+193=?	**a. 288**
82.	Сложить в 7-ой CC165+656=?	**a. 1154**
83.	Что за переменная a?int main(){ . . . int a; . . .}	**a. Автоматическая переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная
84.	Какая функция, команда не используется для динамического выделения памяти?	**a. mem** b. malloc c. calloc d. new
85.	Какая функция, команда не используется для высвобождения динамически выделенной памяти?	**a. destroy** b. delete c. free
86.	Что за переменная a?int main(){ . . . int a; . . .}	**a. Локальная переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная
87.	Что за переменная a?int main(){ . . . static int a; . . .}	**a. Статическая переменная** b. Глобальная переменная c. Динамическая переменная d. Постоянная переменная
88.	Что за переменная a?int a;int main(){ . . .}	**a. Глобальная переменная** b. Локальная переменная c. Динамическая переменная d. Константа
89.	Что делает эта строчка программы: int *p = malloc(400); ?	a. ничего не делает **b. Выделяет 400 байт и записывает адресс выделенного блока в переменную p** c. Выделяет массив из 400 элементов типа int и записывает адресс выделенного блока в переменную p d. Выделяет 400 бит и записывает адресс выделенного блока в переменную p e. free
90.	Какая память является стековой?	**a. Автоматическая**

		b. Глобальная-статическая c. Динамическая
91.	В какой библиотеке описаны функции free и malloc?	**a. stdlib.h** b. conio.h c. windows.h d. DSL.h
92.	какая переменная будет находиться на дне стека в момент исполнения программы в точке ТУТ?int a; int main(){int b;int c;{int d;+++ ТУТ +++} }	a. a **b. b** c. c d. d
93.	какая переменная будет находиться на вершине стека?

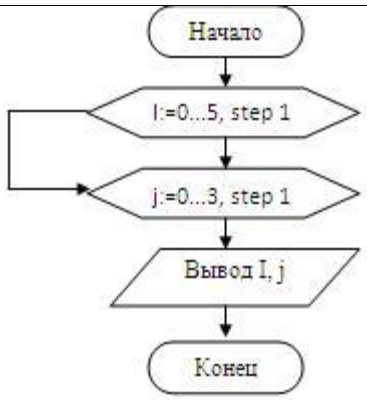
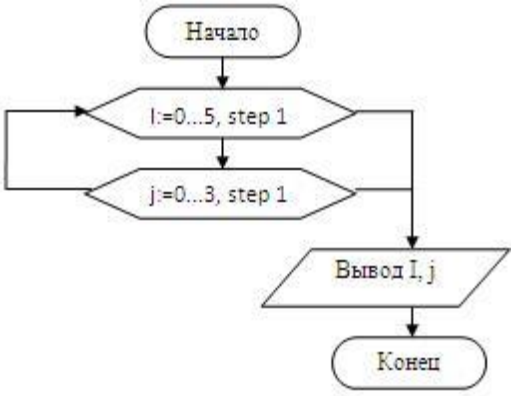
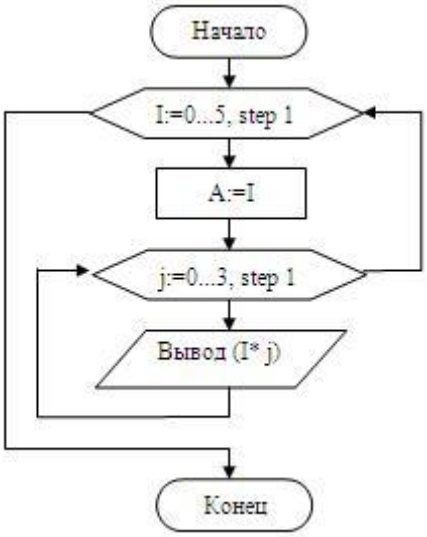
int a;int main(){int b;int c; {int d; +++ ТУТ +++ } } | a. a
b. b
c. c
d. d |
| 94. | Сколько байт выделится с следующей строчке:p = malloc(100*sizeof(int)); | **a. 400**
b. 100
c. 300
d. 800
e. 50 |
| 95. |

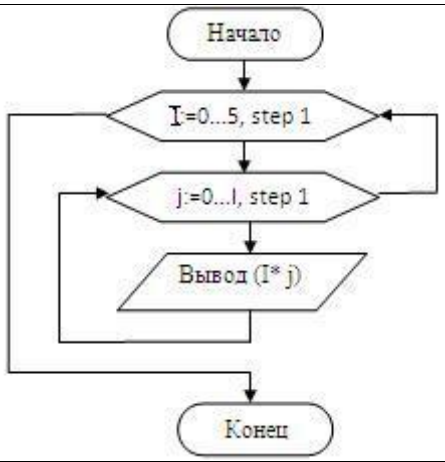
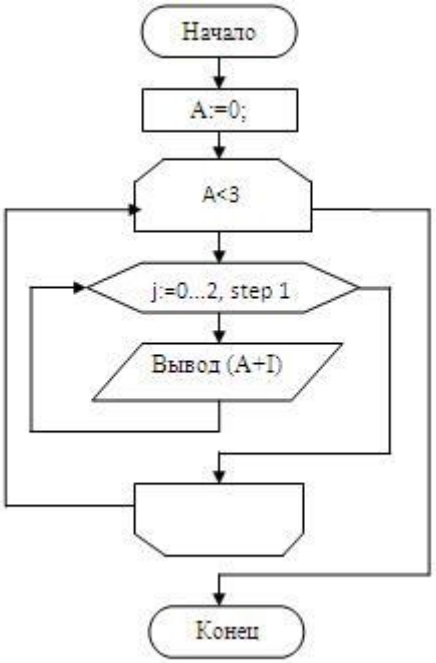
какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ?int a; int main(){ int b; int c; { int d; } --- ТУТpintf …} | a. a
b. b
c. c
d. d |
| 96. | Сколько байт выделится с следующей строчке:p = malloc(100*sizeof(char)); | a. 400
b. 100
c. 300
d. 800
e. 50 |
| 97. | Какое значение вернет функция malloc или new если не сможет выделить память? | **a. NULL**
b. error
c. -1
d. 15 |
| 98. | какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ

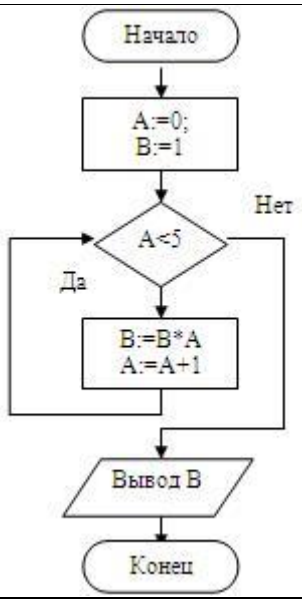
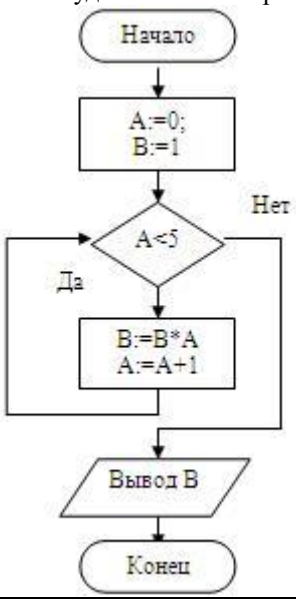
 Что будет выведено на экран в результате работы программы #include <stdio.h>#include <stdlib.h> int f(){ static int a=0; a=a+1; printf("%i",a);} int main(){ f(); f(); f(); system("pause"); return 0;} | **a. 123**
b. 12
c. 111
d. 11
e. 000 |
| 99. | Что такое NULL? | **a. Константа со значением 0**
b. Переменная со значением 0
c. Константа со значением -1
d. Текстовая строка |
| 100. | какая переменная будет находиться на вершине стека в момент исполнения программы в точке ТУТ

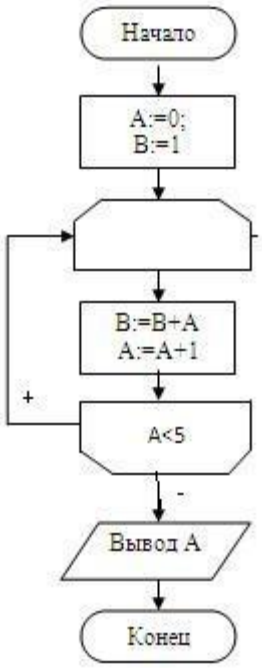
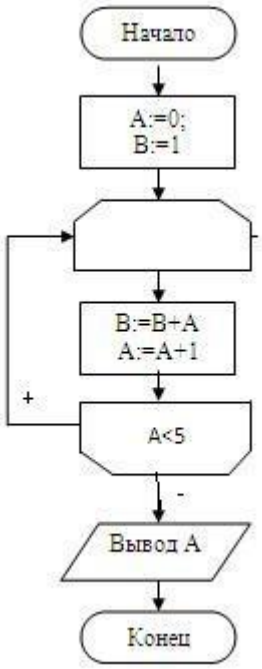
 Что будет выведено на экран в результате работы программы #include <stdio.h>#include <stdlib.h> int f(){ int a=0; a=a+1; printf("%i",a);} int main(){ f(); f(); f(); system("pause"); return 0;} | a. 123
b. 12
c. 111
d. 11
e. 000 |

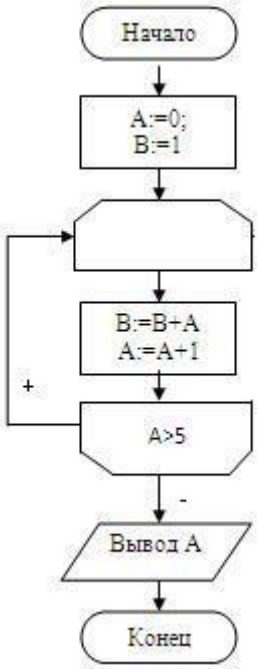
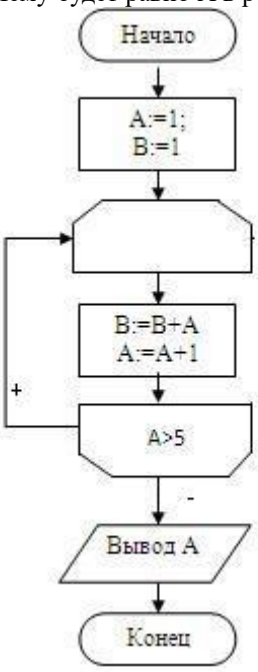
101.	что означает NULL?	**a. пустой указатель** b. указатель на стек c. константа, показывающая ошибку при очистке памяти
102.	В какой момент произойдет выделение памяти под глобальную переменную?	**a. В момент запуска программы** b. при входе в функцию, где переменная используется c. При первом обращении к переменной
103.	int *p = malloc(400);Можно ли проверить по значению указателя, произошла ли ошибка при выделении памяти?	**a. ДА** b. НЕТ
104.	В какой момент произойдет выделение памяти под переменную static?	**a. В момент запуска программы** b. при входе в функцию, где переменная используется c. При первом обращении к переменной d. При инициализации переменной
105.	int main(){ int *p = malloc(400); *(p+1)=4; free(p);}есть ли тут ошибка?	**a. НЕТ** b. ДА
106.	В какой момент произойдет выделение памяти под автоматическую переменную?	a. В момент запуска программы b. при входе в функцию, где переменная используется c. При первом обращении к переменной d. При инициализации переменной **e. В точке объявления переменной**
107.	В отличие от поименованных переменных динамические переменные ...	**a. Не удаляются при выходже из блока** b. НЕТ ОЛИЧИЙ ОТ ПОИМЕНОВАННЫХ ПЕРЕМЕННЫХ c. хранятся в стеково памяти d. хранятся внутри кода программы в памяти
108.	В какой момент произойдет удаление памяти из под глобальной переменной?	**a. В момент завершения программы** b. После момента, когда переменная больше не используется c. При выходе из функции, где эта переменная используется d. Не удаляется никогда, занимая место в оперативной памяти
109.	В какой момент произойдет удаление памяти из под автоматической переменной?	a. В момент завершения программы b. только привыходе из функции где она объявлена **c. При выходе из блока, где переменная объявлена** d. При выходе из любой функции, неважно где переменная объявлена
110.	Есть ли ошибки?Рассматриваем явные ошибки (ошибки в написании алгоритмических структур, с записью БСА, ошибки	**a. ДА** b. НЕТ c. НЕзнаю

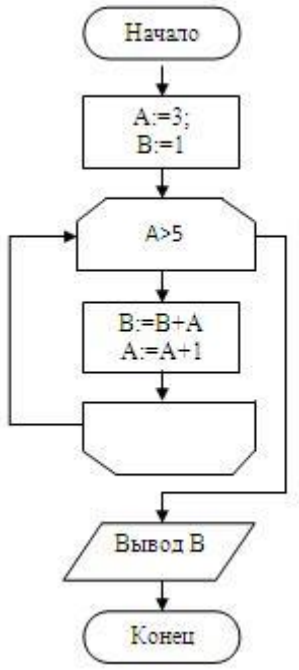
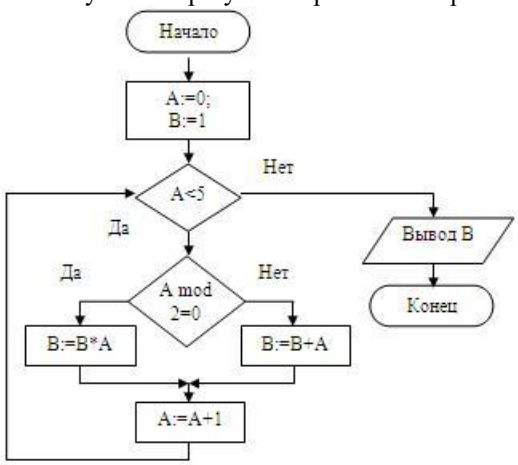
	 использования переменных)	
111.	Есть ли ошибки? Рассматриваем явные ошибки (ошибки в написании алгоритмических структур, с записью БСА, ошибки использования переменных) 	**a. ДА** b. НЕТ c. НЕзнаю
112.	Есть ли ошибки? Рассматриваем явные ошибки (ошибки в написании алгоритмических структур, с записью БСА, ошибки использования переменных) 	a. ДА **b. НЕТ** c. НЕзнаю
113.	Есть ли ошибки? Рассматриваем явные ошибки (ошибки в написании алгоритмических структур, с записью БСА, ошибки использования переменных)	a. ДА **b. НЕТ** c. НЕзнаю

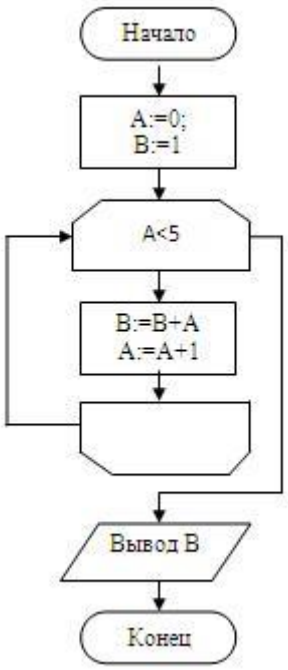
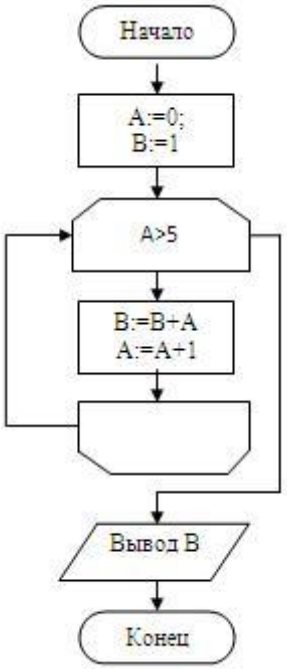
		
114.	Есть ли ошибки? Рассматриваем явные ошибки (ошибки в написании алгоритмических структур, с записью БСА, ошибки использования переменных) 	**a. ДА** b. НЕТ c. НЕзнаю
115.	Какое будет значение переменной А в конце работы алгоритма?	**a. 5**

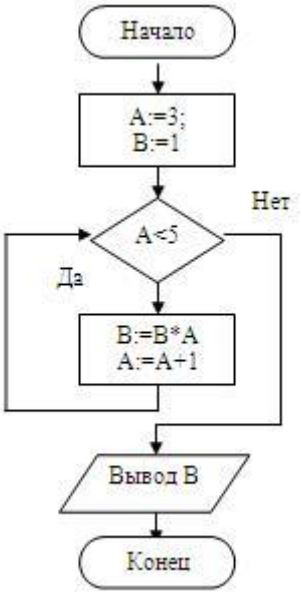
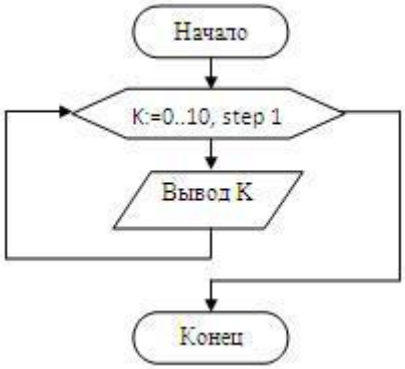
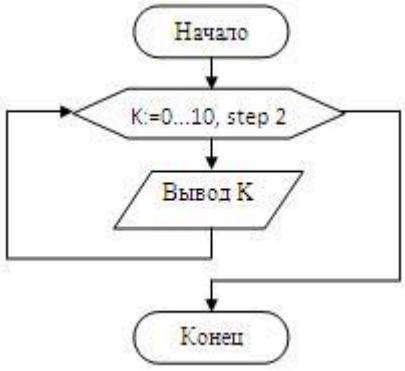
		
116.	Какое будет значение переменной B в конце работы алгоритма? 	**a. 0**
117.	Чему будет равно A в результате работы алгоритма (A=?)?	**a. 5**

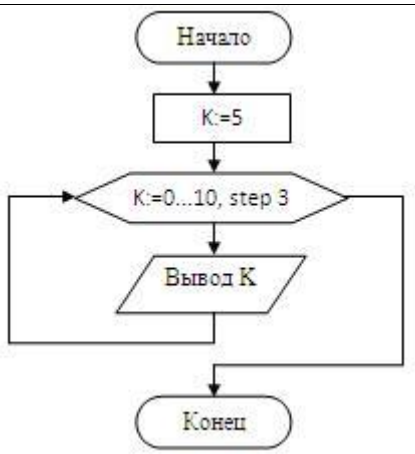
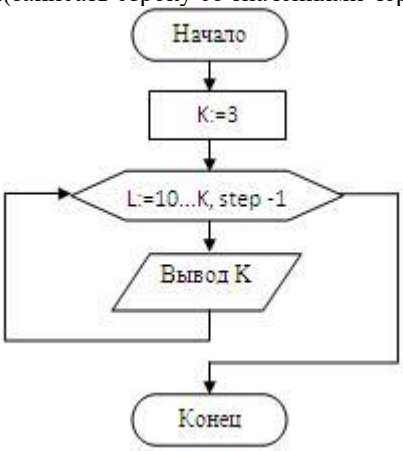
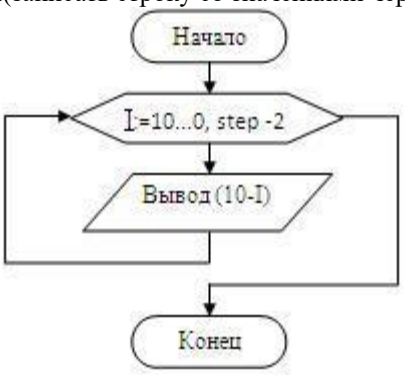
		
118.	Чему будет равно A в результате работы алгоритма (A=?)? 	**a. 5**
119.	Чему будет равно A в результате работы алгоритма (A=?)?	**a. 1**

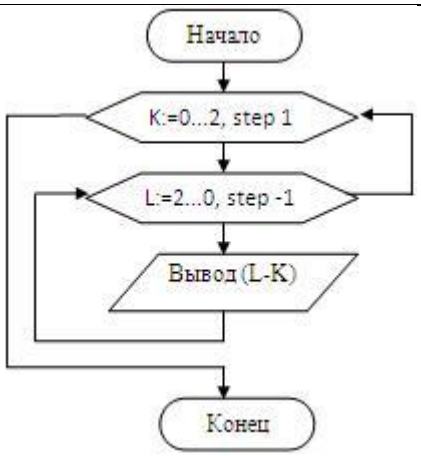
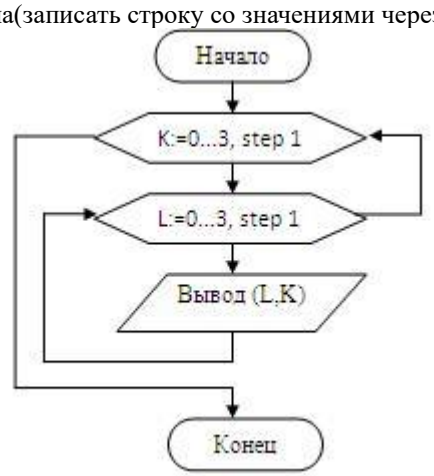
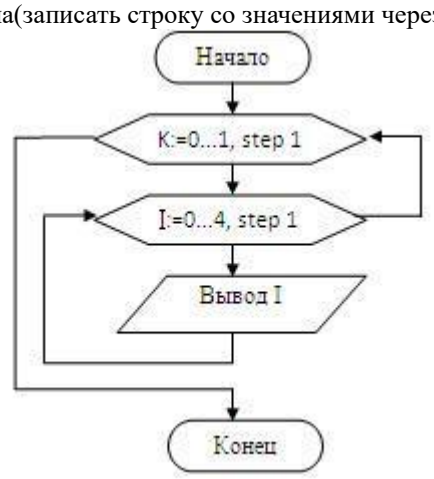
		
120.	Чему будет равно A в результате работы алгоритма (A=?)? 	**a. 2**
121.	Чему будет равно A в результате работы алгоритма (A=?)?	**a. 3**

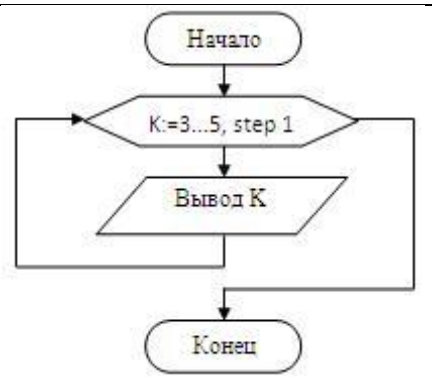
		
122.	Что получится в результате работы алгоритма (B=?)? 	**a. 20**
123.	Что получится в результате работы алгоритма (B=?)?	**a. 11**

		
124.	Что получится в результате работы алгоритма (B=?)? 	**a. 1**
125.	Что получится в результате работы алгоритма (B=?)?	**a. 12**

		
126.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или слитно) 	**a. 0 1 2 3 4 5 6 7 8 9 10** **b. 012345678910**
127.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или слитно) 	**a. 0 2 4 6 8 10** **b. 0246810**
128.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или	**a. 0 3 6 9** **b. 0369**

	 слитно)	
129.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или  слитно)	**a. 3 3 3 3 3 3 3 3** **b. 33333333** **c. 3,3,3,3,3,3,3,3**
130.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или  слитно)	**a. 0 2 4 6 8 10** **b. 0246810** **c. 0,2,4,6,8,10**
131.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или	**a. 2 1 0 1 0 -1 0 -1 -2** **b. 21010-10-1-2**

	 слитно)	
132.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или  слитно)	**a. 00 10 20 30 01 11 21 31 02 12 22 32 03 13 23 33** **b. 0 0 1 0 2 0 3 0 0 1 1 1 2 1 3 1 0 2 1 2 2 2 3 2 0 3 1 3 2 3 3 3** **c. 0,0 1,0 2,0 3,0 0,1 1,1 2,1 3,1 0,2 1,2 2,2 3,2 0,3 1,3 2,3 3,3** d. 0,0,1,0,2,0,3,0,0,1,1,1,2,1,3,1,0,2,1,2,2,2,3,2,0,3,1,3,2,3,3,3 **e. 00102030011121310212223203132333**
133.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или  слитно)	**a. 0 1 2 3 4 0 1 2 3 4** **b. 0123401234**
134.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или	**a. 3 4 5** **b. 345**

	 слитно)	
135.	Что будет выведено на экран в результате работы алгоритма(записать строку со значениями через пробел или  слитно)	**a. 5 4 3 2 1 0** **b. 543210**
136.	Какие из типов являются целочисленными:	**a. int** b. float **c. char** d. long double
137.	Что будет напечатано на экран кодом: <!-- /* Font Definitions */ @font-face {font-family:"Cambria Math"; panose-1:2 4 5 3 5 4 6 3 2 4; mso-font-charset:1; mso-generic-font-family:roman; mso-font-format:other; mso-font-pitch:variable; mso-font-signature:0 0 0 0 0 0;} @font-face {font-family:Calibri; panose-1:2 15 5 2 2 2 4 3 2 4; mso-font-charset:204; mso-generic-font-family:swiss; mso-font-pitch:variable; mso-font-signature:-1610611985 1073750139 0 0 159 0;} /* Style Definitions */ p.MsoNormal, li.MsoNormal, div.MsoNormal {mso-style-unhide:no; mso-style-qformat:yes; mso-style-parent:""; margin-top:0cm; margin-right:0cm; margin-bottom:10.0pt; margin-left:0cm; line-height:115%; mso-pagination:widow-orphan; font-size:11.0pt; font-family:"Calibri","sans-serif"; mso-ascii-font-family:Calibri; mso-ascii-theme-font:minor-latin; mso-fareast-font-family:Calibri; mso-fareast-theme-font:minor-latin; mso-hansi-font-family:Calibri; mso-hansi-theme-font:minor-latin; mso-bidi-font-family:"Times New Roman"; mso-bidi-theme-font:minor-bidi; mso-fareast-language:EN-US;} .MsoChpDefault {mso-style-type:export-only; mso-default-props:yes; mso-ascii-font-family:Calibri; mso-ascii-theme-font:minor-latin; mso-fareast-font-family:Calibri; mso-fareast-theme-font:minor-latin; mso-hansi-font-family:Calibri; mso-	**a. 2** b. 2.5 c. В коде ошибка

	hansi-theme-font:minor-latin; mso-bidi-font-family:"Times New Roman"; mso-bidi-theme-font:minor-bidi; mso-fareast-language:EN-US;} .MsoPapDefault {mso-style-type:export-only; margin-bottom:10.0pt; line-height:115%;} @page Section1 {size:612.0pt 792.0pt; margin:2.0cm 42.5pt 2.0cm 3.0cm; mso-header-margin:36.0pt; mso-footer-margin:36.0pt; mso-paper-source:0;} div.Section1 {page:Section1;} --&int a = 5;int b = 2;int c = a / b;printf("%d", c);	
138.	Сколько байт в памяти занимаетchar str[40];	a. 80 **b. 40** c. 1
139.	Сколько байт в памяти занимаетfloat list[15];	a. 15 b. 30 **c. 60**
140.	Сколько байт в памяти занимаетdouble m[20];	a. 20 b. 40 **c. 160**
141.	Что будет напечатано на экран кодом:int a = 3;int b = 8;if (a <= b) b = a;else b = b + a;printf("%d", b);	**a. 3** b. 11 c. Ничего. В коде ошибка.
142.	Чем различаются вызовы: printf("Hello");puts("Hello");	a. Функция printf печатает перенос строчки в конце **b. Функция puts печатает перенос строчки в конце** c. Не различаются
143.	Что будет напечатано на экран кодом:int a = 4;int b = 7;if (a > b - a) b = a;else b = b + a;printf("%d", b);	**a. 4** b. 3 c. Ничего. В коде ошибка.
144.	Какой набор инструкций содержит ошибку:char surname[100]; scanf("%s", surname);char surname[100]; scanf("%s", &surname);	a. Первый b. Второй c. Оба содержат **d. Оба не содержат**
145.	Что будет напечатано на экран кодом:int a = -3;int b = 11;if (a * b > b)b = a + b;printf("%d", b);	a. 7 **b. 11** c. Ничего. В коде ошибка.
146.	Какой набор инструкций содержит ошибку:int age; scanf("%d", &age);int age; scanf("%d", age);	a. Первый **b. Второй** c. Оба содержат d. Оба не содержат
147.	Что будет напечатано на экран кодом:int a = 3;int b = -7;if (a - b > b + a){b = a * b;}printf("%d", b);	**a. -21** b. -7 c. Ничего. В коде ошибка.
148.	Что будет напечатано на экран кодом:int a = 3;int b = 5;while (a >= 0){b = b + 2;}printf("%d", b);	a. 5 b. 7 **c. Ничего**

149.	Что будет напечатано на экран кодом:int a = 2;int b = 12;while (a < 0){b = b + 2;}printf("%d", b);	**a. 12** b. 14 c. Ничего
150.	Что будет напечатано на экран кодом:int a = 8;int b = 5;while (a > 0){a = a - 3;b = b + 2;}printf("%d", b);	**a. 11** b. 15 c. Ничего
151.	Что будет напечатано на экран кодом:int a = -3;int b = 4;int c = a / b;printf("%d", c);	**a. 0** b. -1 c. В коде ошибка
152.	Что будет напечатано на экран кодом:int a = 0;int b = 2;while (a < 10){ a = a + 5; b = b * b;}printf("%d", b);	**a. 16** b. 32 c. Ничего
153.	Что будет напечатано на экран кодом:int a = 7;int b = 4;int c = a % b;printf("%d", c);	a. 1 **b. 3** c. В коде ошибка
154.	Программой называется:	**a. Запись алгоритма на формальном языке** b. Запись алгоритма на естественном языке
155.	Что будет напечатано на экран кодом:int a = 9;int b = 4;int c = a % b;printf("%d", c);	a. 2.25 **b. 1** c. В коде ошибка
156.	Пошаговая детализация алгоритма/программы соответствует:	a. Принципам восходящего проектирования **b. Принципам нисходящего проектирования**
157.	Что будет напечатано на экран кодом:int a = 9;float b = 6;float c = a / b;printf("%0.1f" ,c);	a. 1 **b. 1.5** c. В коде ошибка
158.	Что будет напечатано на экран кодом:void func(int var){ var = var + 1;}void main(){ int a = 5; func(a); printf("%d", a);}	**a. 5** b. 6 c. Ничего. В коде ошибка.
159.	Чему соответствует инструкцияint age;	**a. Объявлению** b. Определению c. Инициализации
160.	Что будет напечатано на экран кодом:void func(int *var){ *var = *var + 1;}void main(){ int a = 7; func(&a); printf("%d", a);}	a. 7 **b. 8** c. Ничего. В коде ошибка.
161.	Что будет напечатано на экран кодом:void func(int &var){ *var = *var + 1;}void main(){ int a = 4; func(&a); printf("%d", a);}	a. 4 b. 5 **c. Ничего. В коде ошибка.**
162.	с помощью какой функции stdio.h можно открыть текстовый файл для чтения?	**a. fopen** b. fileopen c. openf

		d. fopenf e. fopenread
163.	какой строчкой мы откроем текстовый файл для чтения	**a. f=fopen("text.txt","rt")** b. f=fopen("text.txt","wt") c. f=fopen("text.txt","rt++") d. f=fopen("text.txt","rb") e. f=fopen("text.txt","tr")
164.	какой строчкой мы откроем текстовый файл для записи	a. f=fopen("text.txt","rt") **b. f=fopen("text.txt","wt")** c. f=fopen("text.txt","rt++") d. f=fopen("text.txt","wb") e. f=fopen("text.txt","tw")
165.	Каким атрибутом передается режим открытия файла, например "rt", в функции fopen?	a. первым **b. вторым** c. третьим d. можно передать хоть 1-ым, хоть 2-ый, хоть 3-им
166.	Каким атрибутом передается имя файла в функции fopen?	**a. первым** b. вторым c. третьим d. можно передать хоть 1-ым, хоть 2-ый, хоть 3-им
167.	что возвращает функция fopen в качестве результата?	a. сам файл b. все данный внутри файла **c. указтаель на блок управления файлом** d. указатель на файл e. целочисленную переменную

База тестовых вопросов для 3 семестра

№ п/п	Формулировка вопроса	Варианты ответов (полужирным шрифтом – верные варианты)
1.	Поставьте в соответствие парадигмы программирования и используемые в них абстракции a. Структурное программирование b. Объектно-ориентированное программирование c. Обобщенное программирование d. Функциональное программирование e. Логическое программирование f. Языково-ориентированное программирование	**a. подпрограммы, процедуры, функции** **b. классы** **c. шаблоны классов** **d. лямбда-выражения** **e. предикаты** **f. предметно-ориентированные языки программирования**
2.	Выберите все языки программирования, которые повлияли на создание объектно-ориентированного языка C++	**a. C** b. Smalltalk **c. Simula** d. Object Pascal e. Java
3.	Какая предшествующая парадигма программирования оказала	**a. Структурная**

	сильное влияние на становление объектно-ориентированной парадигмы?	**парадигма программирования** b. Парадигма обобщенного программирования c. Парадигма логического программирования d. Аспектно-ориентированная парадигма e. Объектно-ориентированная парадигма была первой парадигмой программирования в истории
4.	Выберите все парадигмы и подходы к программированию, которые предполагают описание в той или иной форме спецификации решения задачи и указания ожидаемого результата выполнения программы без описания алгоритма решения задачи.	a. Структурное программирование b. Объектно-ориентированное программирование c. Императивное программирование **d. Декларативное программирование** **e. Логическое программирование** **f. Функциональное программирование**
5.	Выберите ключевые недостатки разработки программ на языке ассемблера.	**a. Сложность отладки большого количества строк программного кода** b. Низкая производительность программ на ассемблере **c. Отсутствие переносимости на уровне исходного кода** d. Избыточное потребление оперативной памяти e. Неудобство применения перфокарт и перфорированных лент
6.	Выберите языки программирования, которые реализуют объектно-ориентированную парадигму	a. C **b. C++** **c. Java** d. Pascal **e. Python** f. Prolog
7.	Назовите исторически первый объектно-ориентированный язык	**a. Simula**

	программирования	b. Smalltalk c. C++ d. Objective C e. Object Pascal
8.	Структурная парадигма программирования предполагает отказ от использования в программе ...	**a. операторов безусловного перехода** b. операторов условного перехода c. операторов циклов d. рекурсивного вызова функций e. примитивных типов данных
9.	Даны следующие классы: `public class Test { int testInt = 5; public int getInt() { return testInt; } } class Another { // тело класса }` Укажите все способы получения значения testInt из тела класса Another. (выберите один или несколько правильных вариантов ответа)	a. new Test().testInt b. Test.testInt **c. new Test().getInt()** d. Test.getInt() e. Ничего из перечисленного
10.	Среди перечисленных конструкций C# укажите объявление свойства	a. string GetName() {return "Name";} b. string Name; **c. string Name {get{return "Name";}};** d. string this[int i] {get{return "Name";}};
11.	Динамический метод можно вызвать только в контексте объекта (экземпляра класса)	**a. Верно** b. Неверно
12.	Для каких элементов класса справедливо утверждение: Чем больше в классе этих элементов, тем больше места в памяти занимает каждый экземпляр этого класса (выберите один или несколько правильных вариантов ответа)	**a. string name;** b. string GetName() {return "Name";} c. static string path; d. static string GetPath() {return path;}
13.	Отметьте все корректные обращения к полям объявленного класса SomeClass: class SomeClass { public static int s; public int d; }	**a. SomeClass.s = 42** b. SomeClass.d = 42 c. new SomeClass().s = 42 **d. new SomeClass().d = 42**
14.	Что напечатает код? class SomeClass { public static int s; public int d; } Main () { var obj1 = new SomeClass(); var obj2 = new SomeClass(); obj1.d = 42; obj2.d = 43; Console. Write(obj1.d + " " + obj2.d); }	**a. 42 43**

15.	``` class SomeClass { public static int s = 1; public int d = 1; public void Run() { Console.Write(s + " " + d + " "); s++; d++; } public static void Main() { var object1 = new SomeClass(); var object2 = new SomeClass(); object1.Run(); object2.Run(); object1.Run(); } } ``` Что напечатает вызов метода Main в листинге выше?	**a. 1 1 2 1 3 2**
16.	Дан программный код на языке C++ (определение класса опущено):namespace UniversitySpace{ class Student { ... };}Каким будет полное имя класса?	a. UniversitySpace.Student b. Student **c. UniversitySpace::Student** d. UniversitySpace->Student e. UniversitySpace(Student)
17.	Какими должны быть абстракции в рамках модулей?	**a. Абстракции одного модуля должны быть логически связаны друг с другом** b. Абстракции одного модуля должны быть независимы друг от друга c. Один модуль - только одна абстракция d. Абстракции должны максимально сильно взаимодействовать с абстракциями других модулей **e. Абстракции одного модуля должны быть независимы от реализации абстракций других модулей**
18.	Принцип модульности позволяет для программ	**a. Выполнять компиляцию модулей по отдельности** b. Выполнять компоновку модулей по отдельности c. Нет правильного ответа

19.	Что входит в сигнатуру объявления функции в языка C/C++?	**a. тип возвращаемого значения** **b. имя функции** **c. список формальных параметров** d. список фактических параметров e. тело функции
20.	Выберите абстракции, которые должен содержать заголовочный файл (.h) при грамотной программной реализации	**a. Подключения (#include) других заголовочных файлов проекта и библиотек** **b. Объявления заголовков функций** c. Определения тела функций **d. Определения (объявления) классов** e. Реализация методов, конструкторов и деструкторов классов **f. Объявления пространств имён** g. Использование (using) пространств имён h. Подключения (#include) связанных файлов с исходным кодом (.cpp)
21.	Какие директивы препроцессора могут использоваться для реализации стражей включения в C++?	**a. #ifndef #endif** b. #error **c. #define** **d. #pragma once** e. #warning f. #if #else
22.	С какой целью написан в данном фрагменте кода оператор throw? try{ int* a = new int[10000]; }catch(...){ cout << "Проблемы с выделением памяти - 10000 [int]";throw; // зачем?}	a. Оператор catch обязательно должен завершаться оператором throw; b. Данный код некорректен - ошибка компиляции **c. Для повторного возбуждения исключения с целью передачи его на следующий уровень обработки** d. Для повторной попытки выделения динамической памяти под массив e. Для освобождения

		выделенной динамической памяти
23.	Выберите все допустимые варианты оператора throw в блоке try для функции foo, генерирующие исключения, которые сможет перехватить обработчик catch:void foo(int i, string str, Exception e){ try{// вариант throw}catch(...) { /*обработчик*/ }}	**a. throw 1;** **b. throw i;** **c. throw 'i';** **d. throw str;** **e. throw e;** f. throw Exception; g. throw ...;
24.	Произойдет ли утечка памяти при выполнении данного фрагмента кода?int* a; try{ a = new int[10000]; throw 1;}catch(...){ cout << "Утечет память или нет?";throw;}delete[] a;	**a. Утечка произойдет** b. Утечки не произойдет, выделенная память будет успешно освобождена c. Ошибка компиляции d. Динамическая память не может быть выделена таким способом
25.	Выберите наиболее верное утверждение:	a. Абстракция позволяет заменить решение одной большой задачи решением серии меньших задач **b. Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов** c. Абстракция позволяет отделить свойства объекта от его поведения d. Абстракция позволяет выстроить иерархические связи между объектами
26.	Какая декомпозиция концентрирует внимание на порядке происходящих действий в системе?	**a. Алгоритмическая декомпозиция** b. Объектно-ориентированная декомпозиция
27.	Что позволяет реализовать свойство сложных систем: "Внутрикомпонентная связь обычно сильнее, чем связь между компонентами" ?	**a. Дает возможность относительно изолированно изучать каждую часть** b. Дает возможность построить иерархию связей между компонентами системы c. Позволяет определить сравнительно небольшое количество различных

		типов элементарных компонентов системы d. Позволяет описать более простую систему, предшествующую существующей e. Дает возможность оставить на усмотрение наблюдателя возможность определять структурные связи в системе
28.	Как декомпозиция позволяет бороться со сложностью систем?	**a. Путем разделения сложной системы на более простые элементы** b. Путем выделения наиболее существенных объектов, их свойств и операций c. Путем упорядочивания абстракций в иерархию d. Путем объединения более простых частей в единое целое
29.	Упорядочивание абстракций по уровням - это ...	**a. Иерархия** b. Абстрагирование c. Декомпозиция d. Модульность e. Инкапсуляция
30.	Какой способ борьбы со сложностью применяется директором компании в процессе определения подчиненности заместителей директора, руководителей отделов и рядовых сотрудников?	a. Абстракция **b. Иерархия** c. Алгоритмическая декомпозиция d. Объектно-ориентированная декомпозиция e. Эволюция
31.	Какой признак сложной системы иллюстрирует следующий пример: "Как правило в крупных компаниях с разветвленной иерархической структурой управления количество различных видов должностей сотрудников в разы меньше, чем количество самих сотрудников компании. Каждое из подразделений компании включает в свое штатное расписание разные вариации из этого небольшого перечня типовых должностей."	a. Сложность часто представляется в виде иерархии b. Выбор, какие компоненты в данной системе считаются элементарными, относительно произволен c. Внутрикомпонентная связь обычно сильнее, чем связь между компонентами **d. Иерархические системы обычно**

		состоят из немногих типов подсистем, по-разному скомбинированных и организованных e. Работающая сложная система является результатом развития работавшей простой системы
32.	Почему программному обеспечению присуща сложность?	a. Существование множества языков программирования **b. Сложность в описании решаемых проблем в процессе автоматизации и требований к разработке** **c. Трудности управления процессом разработки** d. Сложность понимания объктно-ориентированной методологии разработки **e. Трудности достижения необходимой степени гибкости программного обеспечения** **f. Проблема описания поведения дискретной системы и ее элементов** g. Постоянное развитие средств вычислительной техники и технологий программирования
33.	Какие компоненты считаются элементарными при рассмотрении персонального компьютера как сложной технической системы?	a. Электротехнические элементы: транзисторы, диоды, конденсаторы, резисторы и т.д. b. Установленные прикладные программы и сервисы операционной системы c. Оборудование компьютера: системный блок, монитор, клавиатура, мышь и т.д. **d. Выбор элементарных компонентов системы остается на усмотрение наблюдателя**

34.	Какие из приведенных пар классов разумнее связать отношением композиции, а не агрегации или иным отношением?	a. Факультет - студент **b. Факультет - декан** c. Факультет - ректор **d. Сотрудник университета - Занимаемая должность** e. Преподаватель - Студент
35.	Укажите, какую информацию о классе указывают на UML-диаграмме классов.	**a. Имя класса** b. Имена объектов класса **c. Поля класса** **d. Методы класса** **e. Модификаторы видимости** f. Код реализации методов класса g. Значения полей объектов класса
36.	В некоторой предметной области Компания может иметь нескольких Поставщиков или не иметь их вовсе. А Поставщик считается поставщиком только в случае, если он осуществляет поставки хотя бы одной Компании.Укажите правильный вариант указания кратностей для связи "Компания - Поставщик".	**a. Компания [1..*] --- Поставщик [*]** b. Компания [1] --- Поставщик [*] c. Компания [*] --- Поставщик [1..*] d. Компания [1..*] --- Поставщик [1..*] e. Компания [*] --- Поставщик [0..1] f. Компания [1] --- Поставщик [0] g. Компания [*] --- Поставщик [*]
37.	Какой вид отношения между классами содержит семантику "является" ("is-a")	**a. Обобщение** b. Ассоциация c. Композиция d. Агрегация e. Зависимость
38.	Выберите виды отношений, которые имеют семантику "Часть - Целое"	**a. Агрегация** **b. Композиция** c. Обобщение d. Реализация e. Ассоциация f. Зависимость
39.	Какие диаграммы UML предназначены для описания взаимодействия объектов системы во времени?	**a. Диаграммы кооперации** **b. Диаграммы последовательностей** c. Диаграмма деятельности d. Диаграмма классов e. Диаграмма объектов f. Диаграмма

		прецедентов
40.	Какой вид диаграммы на UML аналогичен блок-схеме алгоритма?	**a. диаграмма деятеятельности** b. диаграмма взаимодействия c. диаграмма последовательности d. диаграмма классов e. диаграмма прецедентов
41.	Выберите диаграммы, которые используются в UML для описания поведения системы (динамических аспектов)	**a. диаграммы взаимодействия** **b. диаграммы состояний** **c. диаграммы прецедентов (вариантов использования)** **d. диаграммы деятельности** e. диаграммы классов f. диаграммы развертывания g. диаграммы компонентов
42.	Какой вид диаграммы акцентирует внимание на временной упорядоченности сообщений, использует элементы "фокус управления" и "линия жизни объекта" и дает наглядную картину, позволяющую понять развитие потока управления во времени?	a. Диаграмма прецедентов b. Диаграмма деятельности c. Диаграмма состояний d. Диаграмма кооперации **e. Диаграмма последовательностей**
43.	Какие отношения могут быть между акторами на диаграмме прецедентов?	**a. Обобщения** b. Ассоциации c. Включения (include) d. Расширения (extends) e. Агрегации f. Реализации
44.	Какие отношения могут быть между прецедентами на диаграмме прецедентов?	**a. Обобщения** b. Ассоциации **c. Включения (include)** **d. Расширения (extends)** e. Агрегации f. Зависимости
45.	Какие элементы отображаются на диаграмме сценариев использования (Use cases)	**a. Актор** **b. Прецедент** c. Класс d. Интерфейс e. Линии жизни объекта (object lifeline)

46.	При анализе предметной области, что позволяет выявить вопрос:«Что может делать объект, что с ним можно делать, как объект взаимодействует с другими объектами?»	**a. Операции объекта** b. Индивидуальность объекта c. Классы предметной области d. Атрибуты объекта e. Значения атрибутов объекта f. Нет правильного ответа
47.	Что более полно подходит под определение "Интерфейс"?	a. Устройство двигателя автомобиля b. Особенности поведения автомобиля на скользкой дороге c. Совокупность всех характеристик автомобиля **d. Совокупность органов управления автомобилем (руль, педали, переключатели)**
48.	Какая разница между объектом и классом?	a. Класс - это исходный код, а объект - скомпилированный и выполняемый код b. Класс описывает категорию, к которой могут либо принадлежать, либо не принадлежать объекты данного класса c. Класс может иметь много экземпляров, а объект - один или ниодного d. Класс может инстанциировать объекты, а сам объект - нет **e. Объект - это экземпляр класса**
49.	Что можно сказать об объектах одного класса в рамках предметной области?	a. Объекты одного класса могут иметь разное поведение **b. Объекты одного класса могут иметь разные значения атрибутов** c. Объекты одного класса могут иметь разные наборы атрибутов

		d. Объекты одного класса могут иметь разные множества значений атрибутов **e. Объекты одного класса могут иметь разное время жизни** f. Объекты одного класса могут иметь разные интерфейсы
50.	Согласно определению понятия "объект" от Гради Буча (Grady Booch)"An object has ..."	**a. state** **b. behavior** **c. identity** d. encapsulation e. class f. instance
51.	Выберите наиболее подходящее определение понятию "Класс"	a. Класс описывает поведение некоторой сущности b. Класс содержит набор функций **c. Класс описывает характеристики и поведение объектов** d. Класс характеризует состояние объекта
52.	Выберите все корректные примеры класса и объекта данного класса	**a. Класс: Сотрудник - Объект: Иванов Иван Иванович с табельным номером 1001** b. Класс: Сотрудник - Объект: Менеджер среднего звена c. Класс: Сотрудник - Объект: Должность сотрудника d. Класс: Здание - Объект: Квартира e. Класс: Здание - Объект: Многоэтажный строящийся дом **f. Класс: Здание - Объект: Многоэтажный дом по адресу г. Челябинск, пр. Ленина 75**
53.	Как объектно-ориентированных подход рассматривает предметную область?	**a. Как совокупность взаимодействующих объектов** b. Как совокупность взаимодействующих подпрограмм c. Как совокупность

		взаимодействующих классов d. Как совокупность взаимодействующих интерфейсов e. Как совокупность взаимодействующих обобщенных типов
54.	Какие артефакты в процессе разработки программной системы можно создавать с помощью средств языка UML?	**a. Архитектура системы** **b. Требования к системе** **c. Функциональные тесты** **d. Алгоритмы работы отдельных модулей** e. Программный код
55.	Язык UML можно использовать непосредственно для ...	**a. визуализации абстракций** **b. проектирования архитектуры системы** c. реализации алгоритмов в программном коде **d. документирования проектных решений**
56.	Диаграммы UML позволяют описывать в процессе моделирования ...	a. только структуру системы b. только поведение системы **c. и структуру и поведение системы** d. нет правильного ответа
57.	Процесс перехода от абстракции к модели называется ...	a. Абстрагированием **b. Моделированием** c. Декомпозицией d. Реализацией
58.	Выберите утверждения корректно описывающие принципы моделирования в программной инженерии	**a. В процессе моделирования необходимо обращать внимание на существенные детали в контексте решаемой задачи** **b. В процессе моделирования используется определенная нотация или язык** c. Выбор модели и средств моделирования

		определяется языком моделирования и не должен зависеть от контекста решаемой задачи d. При моделировании сложных систем лучше ограничиться минимально возможным количеством моделей e. Хорошая модель должна описывать все аспекты структуры и поведения системы, все возможные детали и особенности реализации
59.	Кто из перечисленных ниже знаковых личностей для объектно-ориентированной методологии являлся разработчиком первых версий языка UML?	**a. Гради Буч** **b. Джеймс Рамбо** **c. Ивар Якобсон** d. Мартин Фаулер e. Бьёрн Страуструп f. Эрих Гамма
60.	Канонические диаграммы UML подразделяются на	**a. Поведенческие** **b. Структурные** c. Графические d. Группирующие e. Композиционные f. Иерархические g. Аннотирующие
61.	Определена следующая иерархия наследования (реализация классов опущена для упрощения):class Animal { };class Fish : public Animal { };class Mammal : public Animal { };class Human : public Mammal { };Какой тип может иметь переменная, чтобы объект класса Mammal мог быть создан корректно (выберите все варианты):<ТИП> obj = new Mammal();	**a. Animal** **b. Mammal** c. Fish d. Human e. Любой тип приведет к ошибке компиляции
62.	Каким будет результат выполнения следующего кода: using System; public class Program{ public static void Main() { Child obj; obj.Show(2); }} public class Parent{ public void Show(int iToShow) { Console.WriteLine(iToShow); } } public class Child : Parent{ public void Show() { Console.WriteLine(1); }}	a. 1 b. 2 **c. ошибка компиляции** d. ошибка времени выполнения
63.	Чем можно заменить строку "// 1" чтобы программа скомпилировалась. Выберите все правильные варианты ответа: using System; public class Program{ public static void Main() { B b = new B(); b.SomeMethod(); }} public class A{ public void SomeMethod() { /* ... */ }} public class B : A{ public void SomeMethod(int someArgument) { /* ... */ } // 1}	**a. код скомпилируется и так - изменений не требуется** **b. public void SomeMethod() { base.SomeMethod(); }** c. public virtual void SomeMethod(); d. public void A.SomeMethod();

		e. public void B.SomeMethod();
64.	Выберите все верные утверждения о производном классе (подклассе)	**a. Производный класс наследует все поля и методы родительского класса, включая private** **b. Производный класс может иметь дополнительную функциональность** c. Производный класс наследует только public- и protected-методы и поля родительского класса d. Производный класс может наследоваться только от одного родительского класса e. Имя производного класса должно совпадать с именем родительского класса f. Производный класс имеет прямой доступ ко всем унаследованным полям и методам родительского класса, включая private
65.	Выберите наиболее точное определение полиморфизма	a. это механизм, который объединяет данные и методы, манипулирующие этими данными, и защищает и то и другое от внешнего вмешательства или неправильного использования **b. это принцип, согласно которому объекты, имеющие одинаковый интерфейс, могут вести себя по-разному** c. это механизм, позволяющий создавать классы объектов на основе других классов, расширяя и частично изменяя их функциональность и набор атрибутов d. это принцип ООП. согласно которому каждый обьект может

		использоваться более чем в одной программе e. это процесс сокрытия компонентов данных и кода, реализующего функциональность, за интерфейсом, не позволяющим пользователю искажать данные
66.	Допустимо ли определение абстрактного метода следующим образом: public class Abstract{ public abstract void Method() { /* функционал */ }}	a. Да, метод доступен для вызова у объектов по имени Method() b. Да, метод доступен для вызова у класса по имени Abstract.Method() c. Да, но метод не может быть вызван **d. Нет, недопустимо**
67.	Что будет выведено после компиляции и выполнении следующего кода: using System; public class Program{ public static void Main() { B b = new B(); }} public class A{ public A(){ Console.Write("A() "); } ~A(){ Console.Write("~A()"); }} public class B : A{ public B(){ Console.Write("B() "); } ~B(){ Console.Write("~B() "); }}	a. A() B() ~A() ~B() **b. A() B() ~B() ~A()** c. B() A() ~A() ~B() d. B() A() ~B() ~A() e. Ошибка компиляции f. Ошибка времени выполнения
68.	Что будет выведено на экран в результате выполнения данного кода? using System; public class Program{ public static void Main() { B b = new B(); Method(b); } public static void Method(A a) { a.Method(); }} public class A{ public virtual void Method() { Console.Write("A"); }} public class B : A{ public override void Method() { Console.Write("B"); }}	a. A **b. B** c. ошибка компиляции d. ошибка времени выполнения
69.	Объект какого класса невозможно создать?Выберите все возможные варианты:	a. Класса, имеющего хотя бы один виртуальный метод b. Класса, у которого все методы - виртуальные **c. Класса, имеющего хотя бы один чисто виртуальный метод** d. Класса, у которого нет полей e. Класса, у которого отсутствует конструктор с параметрами **f. Класса, который унаследовал чисто виртуальный метод у суперкласса и не переопределил его**
70.	У класса определены два перегруженных	**a. MyClass obj;**

	конструктора:MyClass::MyClass();MyClass::MyClass(int a, int b);Выберите все строки кода, при выполнении которых происходит вызов конструктора класса	**b. MyClass obj(50,100);** **c. MyClass obj[10];** **d. MyClass* obj = new MyClass;** e. MyClass* obj = new MyClass(50); f. obj->MyClass(); g. obj->MyClass(50,100);
71.	Выберите все варианты с корректно компилируемой реализацией метода Count класса Counterclass Counter{int num;public: void Count();};//выберите варианты реализации метода вне объявления класса	a. void Counter::Count(); **b. void Counter::Count() {}** **c. void Counter::Count() {num++;}** d. void Count() {count++;} e. void Counter::Count() = 0; f. void Counter::Count(int num) {num++;} g. Counter::Count() {num++;}
72.	Выберите все элементы класса, которые не могут иметь типа возвращаемого значения (даже void)	**a. Конструктор** **b. Деструктор** c. Статический метод d. Нестатический метод
73.	В некотором блоке кода создан объект в динамической памяти:{MyClass* obj = new MyClass;// использование obj// строка 1}// строка 2Выберите в какой строке и с помощью какого кода необходимо произвести удаление объекта из динамической памяти (если это действительно необходимо и не произойдет автоматически).	**a. В строке 1: delete obj;** b. В строке 1: delete* obj; c. В строке 1: delete[] obj; d. В строке 2: delete obj; e. В строке 2: delete* obj; f. В строке 2: delete[] obj; g. Объект при выходе из блока кода будет удален из динамической памяти автоматически
74.	Выберите все подходящие варианты для создания объекта и вызова метода у класса MyClassclass MyClass{ public: void count();}Варианты ответов:1) MyClass *obj = new MyClass; obj->count();2) MyClass obj = new MyClass; obj.count();3) MyClass* obj = new MyClass; obj.count();4) MyClass* obj = new MyClass; (*obj).count();5) MyClass obj = new MyClass; (*obj).count();	**a. 1** b. 2 c. 3 **d. 4** e. 5
75.	Выберите все корректные варианты строки инициализации поля number класса MyClass в конструкторе:class MyClass{ int number;public: MyClass::MyClass(int num)};MyClass::MyClass(int num){ //выберите варианты...}	a. *number = *num; **b. this->number = num;** c. this->number = this->num; d. this.number = num; **e. (*this).number = num;** **f. number = num;**
76.	Уничтожение объекта, созданного в статической (глобальной) области памяти в C++ ...	**a. происходит во время выхода из функции main**

		b. происходит во время выхода из блока кода c. происходит во время вызова оператора delete d. объекты в глобальной памяти не уничтожаются e. объекты в глобальной памяти в C++ не могут быть созданы
77.	Выберите все верные утверждения про статические и нестатические методы класса	**a. Статические методы могут работать только со статическими полями класса** b. Нестатические методы могут работать только с нестатическими полями класса **c. Статические методы класса можно вызвать без указания объекта** d. Нестатические методы класса можно вызвать без указания объекта **e. Статические методы класса можно вызвать с указанием объекта класса** f. Статические методы класса не могут возвращать значения (даже void)
78.	Дан следующий фрагмент кода:MyClass* pA = new MyClass;MyClass* pB = new MyClass;// строка 1pA = pB;// строка 2delete pA;Возникает ли в данном коде утечка динамической памяти? Если да, то как её устранить?	**a. устранить утечку памяти - строка 1: delete pA;** b. устранить утечку памяти - строка 2: delete pA; c. устранить утечку памяти - строка 1: delete pB; d. устранить утечку памяти - строка 2: delete pB; e. утечки памяти не возникает f. в коде допущены ошибки и он не может быть скомпилирован
79.	Что описывает понятие "контракт" в ООП?	**a. Спецификацию интерфейса класса (системы, модуля)** b. Особенности внутренней реализации класса (системы, модуля)

		c. Особенности структуры класса (системы, модуля) d. Множество значений атрибутов класса (системы, модуля) e. Перечень объектов класса (экземпляров системы, модулей)
80.	Выберите модификаторы доступа, которые используются в языке C#	**a. private** **b. protected** **c. public** d. abstract e. virtual f. static **g. internal**
81.	Какое из определений описывает понятие "интерфейс" ?	**a. Абстракция поведения класса** b. Механизм, реализующий поведение класса c. Инкапсуляция поведения класса d. Иерархия поведения класса e. Методы и поля класса
82.	Выберите вариант наиболее полно описывающий цель инкапсуляции	**a. Обеспечить независимость внутренней реализации объектов (классов) от остальных частей системы** b. Защитить внутреннюю структуру классов от несанкционированного доступа c. Обеспечить компактность и логическую завершенность объекта (класса) d. Запретить создание объектов класса в динамической области памяти e. Обеспечить независимость поведения объекта класса от внутренней реализации методов класса
83.	Изменения каких элементов класса при правильном соблюдении требований инкапсуляции НЕ должны затронуть	**a. Реализация private-методов**

	другие классы системы?	**b. Сигнатура private-методов** c. Сигнатура public-методов **d. Реализация public-методов** e. Тип и имя public-поля **f. Тип и имя private-поля**
84.	Определите, какие элементы класса "Автомобиль" стоит скрыть согласно требованиям инкапсуляции?	**a. Устройство двигателя** b. Руль, педали газа и тормоза, рычаг переключения передач **c. Устройство коробки переключения передач** d. Интерфейс бортового компьютера **e. Программную прошивку бортового компьютера**
85.	Выберите типы контейнеров STL, в которых можно хранить несколько экземпляров одного и того же значения (замечание: для map - хранить несколько повторяющихся ключей).	**a. vector** **b. deque** **c. list** d. map e. set **f. multimap** **g. multiset**
86.	Выберите типы контейнеров STL, которые сохраняют порядок вставки элементов (т.е. итераторы извлекают значения в том порядке, в котором они были вставлены пользователем).	**a. vector** **b. deque** **c. list** d. map e. set f. multimap g. multiset
87.	В каком случае зайдействован механизм абстракции:	a. Кошка - это подвид семейства кошачьих отряда хищников b. Кошка - это наш домашний любимец Пушок **c. Кошка - это хвостатое существо, умеющее ловить мышей**
88.	Какому классу может соответствовать объект "Боинг 737":	**a. Боинг 737** **b. Самолет** **c. Самолет модели "Боинг"** **d. Воздушное судно** e. Корабль

89.	Что можно назвать классом?	**a. Медведь** b. Кошка по кличке "Пушок" c. Стол с царапиной на столешнице **d. Стол** **e. Средство передвижения**
90.	Что можно назвать экземпляром класса?	a. Медведь **b. Кошка по кличке "Пушок"** **c. Стол с царапиной на столешнице** d. Стол e. Средство передвижения
91.	Выберите наиболее верное определение:	a. ООП - это образ мышления программиста при работе со сложными системами b. ООП - парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних **c. ООП — это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.** d. ООП - методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков
92.	Выберите публичный модификатор доступа:	a. # **b. +** c. -

93.	Выберите защищенный модификатор доступа:	**a. #** b. + c. -
94.	Что такое инкапсуляция?	**a. Инкапсуляция — это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение** b. Инкапсуляция — это методология проектирования c. Инкапсуляция — это экземпляр класса d. Инкапсуляция — это семантическая и синтаксическая конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом
95.	Если вам нужно починить автомобиль, то вы думаете о нем, как о наборе деталей. Это пример:	**a. Декомпозиция** b. Полиморфизм c. Инкапсуляция d. Наследование
96.	Чтобы решить задачу, вы разделяете ее на совокупность более простых задачь. Это пример:	**a. Декомпозиция** b. Полиморфизм c. Инкапсуляция d. Наследование
97.	Если рассматривать зеленый танк в контексте ООП, то это:	**a. Объект** b. Класс c. Абстрактный класс d. Абстрактный объект
98.	В чем разница между экземпляром класса и объектом?	**a. Это эквивалентные термины** b. Объект - это прототип класса c. Класс - это наследник объекта d. Класс инкапсулирует объект
99.	Если рассматривать чертежи танка в контексте ООП, то это:	**a. Класс** b. Объект c. Абстрактный класс d. Абстрактный объект
100.	Если рассматривать понятие "Человек" в контексте ООП, то	**a. Класс**

	это:	b. Объект c. Абстрактный класс d. Абстрактный объект
101.	Чем являет класс Человек для класса Студент?	**a. Родительским классом** b. Базовым объектом c. Родительским объектом
102.	Если рассматривать класс Человек с точки зрения изготовителя перчаток, то тот факт, что у человека 5 пальцев на руке, является реализацией:	**a. Интерфейс** b. Полиморфизм c. Абстрактный класс d. Наследование
103.	Человек произошел от неандертальца, неандерталец от homo erectus, homo erectus от парантропа и т.д. Это пример:	**a. Ирерархия классов** b. Полиморфизм c. Абстрагирование d. Генетический алгоритм
104.	Какие композиции используются в ООП:	a. Объектно-ориентированная b. Алгоритмическая **c. Обе**
105.	Для чего нужна инкапсуляция?	**a. Инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.** b. Инкапсуляция служит для того, чтобы описать набор функций. c. Инкапсуляция служит для того, чтобы разделить задачу на совокупность более простых задачь.
106.	Выберите наиболее верное определение:	**a. Интерфейс — это семантическая и синтаксическая конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом.** b. Интерфейс — это процесс отделения друг от друга элементов объекта, определяющих

		его устройство и поведение c. Интерфейс — это методология проектирования
107.	Чем отличается контракт класса от его интерфейса?	**a. Ничем. Это эквивалентные понятия.** b. Это два противоположных понятия. c. Интерфейс - это частный случай контракта класса.
108.	В каких случаях нужно использовать свойства класса?	**a. Для предоставления доступа к состоянию класса** b. Для сокрытия данных класса c. Для реализации контракта класса
109.	Может ли свойство класса реализовывать поведение класса?	**a. Нет** b. Да c. В некоторых случаях
110.	Может ли класс состоять из одних только свойств?	**a. Да** b. Нет c. Иногда
111.	В каких случаях нужно использовать методы класса?	**a. Для выражения поведения класса** b. Для предоставления доступа к состоянию класса
112.	Может ли метод класса хранить состояние класса?	**a. Нет** b. Да c. В некоторых случаях
113.	Может ли класс состоять из одних только методов?	**a. Да** b. Нет c. Иногда
114.	В каких случаях нужно использовать поля класса?	**a. Для хранения состояния класса** b. Для реализации контракта класса c. Для реализации поведения класса
115.	Что такое поведение класса?	**a. Поведение - это активность объекта, служащая для**

		осуществления контакта с окружающим миром. b. Поведение - это совокупность данных класса c. Поведение - это совокупность данных объекта d. Поведение - это изменение свойств и методов классса
116.	Где в реальном мире встречается инкапсуляция?	**a. Повсеместно** b. Только в программировании c. Только в моделировании предметных областей d. Исключительно в умах программистов
117.	Обязательно ли функция класса должна возвращать значение?	**a. Не обязательно** b. Обязательно
118.	Как называется функция, которая не возвращает значений?	**a. Функция** b. Метод c. Процедура d. Свойство
119.	Что такое состояние класса?	**a. Состояние - это совокупность значений свойств и полей класса в конкретный момент времени.** b. Состояние - это совокупность всех методов класса c. Состояние - это совокупность всех методов и свойств класса
120.	Может ли состояние класса влиять на поведение класса?	**a. Да** b. Нет c. Никогда
121.	Может ли поведение влиять на состояние класса?	**a. Да** b. Нет c. Всегда d. Никогда
122.	Может ли функция класса не влиять на состояние класса?	**a. Да, может** b. Нет, не может
123.	Каким средствами выражается состояние класса?	**a. Свойства** b. Методы

		c. Интерфейсы
124.	Каким средствами выражается состояние класса?	**a. Поля** b. Методы c. Интерфейсы
125.	Должна ли внутри свойства класса выполняться сложная функция?	**a. Нет** b. Да c. Иногда
126.	Может ли доступ к свойству ограничиваться модификаторами видимости?	**a. Да** b. Нет
127.	Вы знаете, что для того, чтобы включить компьютер, нужно нажать на большую кнопку. Это пример:	**a. Инкапсуляция** b. Наследование c. Полиморфизм d. Рефлекс
128.	Какими средствами выражается поведение класса?	**a. Методы** b. Свойста c. Поля d. Переменные
129.	Может ли доступ к функции ограничиваться модификаторами видимости?	**a. Да** b. Нет c. Эти понятия не связанны
130.	Если функция только присваевает скрытому полю класса значение, переданное ей в качестве аргумента, то...	**a. ... она берет на себя роль автоматического свойства** b. ... она берет на себя роль метода c. ... она берет на себя роль свойства
131.	Если функция только возвращает значение скрытого поля класса, то...	**a. ... она берет на себя роль свойства** b. ... она берет на себя роль метода c. ... она берет на себя роль автоматического свойства
132.	Класс может выводить приветствие на экран, это:	**a. Поведение** b. Состояние
133.	Класс может подсчитывать баланс предприятия, это:	**a. Поведение** b. Состояние
134.	У класса "Человек" есть информация о его возрасте, это:	**a. Состояние** b. Поведение
135.	Зачем нужны автоматические свойства?	**a. Чтобы не заводить поля класса, если они не нужны.**

		b. Чтобы определять поведение класса. c. Чтобы гарантировать выполнение контракта класса.
136.	Могут ли автоматические свойства реализовать поведение класса?	**a. Нет** b. Да
137.	Количество чернил в классе "Ручка" - это	**a. Состояние класса** b. Поведение класса c. Контракт класса
138.	Чтобы включить любой компьютер, нужно нажать на кнопку, это пример:	**a. Наследование** b. Полиморфизм c. Инкапсуляция d. Рефлекс
139.	Класс "Точилка" переводит свойство "Острый" класса "Карандаш" в состояние "Заточен" -	**a. Это поведение класса "Точилка"** b. Это поведение класса "Карандаш" c. Это состояние класса "Точилка" d. Это состояние класса "Карандаш"
140.	Реализуют ли автоматические свойства поведение класса?	**a. Нет** b. Да
141.	Может ли доступ к автоматическим свойствам ограничиваться модификаторами видимости?	**a. Да** b. Нет
142.	Чем вы воспользуетесь если вам нужно предоставить доступ без ограничений и контроля к состоянию класса?	**a. Автоматическими свойствами** b. Свойствами c. Методами
143.	Чем вы воспользуетесь если вам нужно контролировать доступ к состоянию класса?	**a. Свойствами** b. Автоматическими свойствами c. Методами
144.	Как зависят состояния разных экземпляров одного класса?	**a. Они не зависят друг от друга** b. Пропорционально c. Если меняется состояние одного экземпляра, то и меняется состояние другого
145.	Может ли класс не иметь ни свойств, ни методов?	**a. Да** b. Нет
146.	Может ли отличаться поведение двух разных экземпляров	**a. Да, может**

	одного класса?	b. Нет, не может
147.	Люди произошли от обезьян, это пример:	**a. Наследование** b. Инкапсуляция c. Полиморфизм d. Генетический алгоритм
148.	У всех легковых автомобилей 4 колеса. Это пример:	**a. Наследование** b. Полиморфизм c. Абстрагирование d. Абстрактный метод
149.	Когда вы едете на автомобиле, вы не задумываетесь о давлении в шинах, это пример:	**a. Абстрация** b. Полиморфизм c. Наследование d. Невнимательность
150.	Вы не задумываетесь об устройстве телефона, когда разговариваете по нему. Это пример:	**a. Абстрация** b. Полиморфизм c. Инкапсуляция d. Наследование
151.	Какие механизмы в ОО языках обычно позволяют обеспечить инкапсуляцию объектов?	**a. Модификаторы доступа** b. Виртуальные методы c. Обработка исключений d. Динамическое выделение памяти e. Статические методы
152.	Как называется способность объекта скрывать свои данные и реализацию от других объектов системы?	a. Полиморфизм **b. Инкапсуляция** c. Абстракция d. Наследование
153.	Выберите наиболее точный пример инкапсуляции:	**a. Вы не знаете как устроен внутри телевизор, но знаете как пользоваться пультом дистанционного управления чтобы переключать каналы** b. У отца с рыжими волосами родился сын с такими же рыжими волосами c. Вы надували воздушный шарик и он лопнул
154.	Что класс-потомок наследует от класса-предка?	a. Только свойства b. Только методы c. Только поля **d. Все члены класса**

155.	Может ли класс-потомок изменить поведение методов, унаследованных от класса-предка?	**a. Может** b. Не может
156.	Может ли класс-потомок иметь функции, которых нет у класса-предка?	**a. Может** b. Не может
157.	У людей есть противопоставленный палец, как и у обезьян. С точки зрения ООП, примером чего является данная ситуация (выберите наиболее точный вариант)?	**a. Наследование** b. Композиция c. Инкапсуляция d. Полиморфизм
158.	В чужой библиотеке классов, которую вы не можете редактировать, есть класс A, который полностью вас устраивает за исключением одного метода. Какой механизм ООП вы примените для решения данной проблемы?	**a. Наследование** b. Композиция c. Инкапсуляция d. Полиморфизм
159.	Выберите наиболее точный пример полиморфизма:	**a. Вы решили покормить птицу на улице. Вам все равно какого вида эта птица, вам достаточно того, что она ведет себя так же как и другие птицы.** b. Вы надували воздушный шарик и он лопнул c. У отца с рыжими волосами родился сын с такими же рыжими волосами d. Вы не знаете как устроен внутри телевизор, но знаете как пользоваться пультом дистанционного управления чтобы переключать каналы
160.	Если вы создали наследника класса, но его поведение не меняли, является ли это примером полиморфизма?	a. Да **b. Нет**

4. Порядок проведения и критерии оценивания промежуточной аттестации

4.1.Порядок проведения промежуточной аттестации

Курс 1, Семестр 2:

Экзамен проводится в виде тестирования. Студент должен ответить на вопросы закрытого типа, которые предполагают выбор вариантов ответа, а также на вопросы открытого типа, которые не предполагают вариантов ответа, правильный ответ требуется написать самостоятельно. Всего 20 тестовых вопросов. Продолжительность теста – 35 минут.

Курс 2, Семестр 3:

Экзамен проводится в виде тестирования. Студент должен ответить на вопросы закрытого типа, которые предполагают выбор вариантов ответа, а также на вопросы открытого типа, которые не предполагают вариантов ответа, правильный ответ требуется написать самостоятельно. Всего 20 тестовых вопросов. Продолжительность теста – 35 минут.

4.2.Критерии оценивания промежуточной аттестации по видам оценочных средств

4.2.1. Критерии оценивания теста

Тест формируется в системе электронного обучения MOODLE. Максимальный балл за тест — 100 баллов.

Оценка	Отлично/ Зачтено	Хорошо/ зачтено	Удовлетворительно/зачтено	Неудовлетворительно/ незачтено
Баллы	100-90 баллов	89-75 баллов	74-60 баллов	59-0 баллов
Уровень освоения проверяемых компетенций	высокий	средний	базовый	недостаточный

4.3.Результаты промежуточной аттестации и уровни сформированности компетенций

При подведении итогов учитываются результаты только промежуточной аттестации:

0-59 баллов – неудовлетворительно/незачтено;
60-74 баллов – удовлетворительно/зачтено;
75-89 баллов – хорошо/зачтено;
90-100 баллов – отлично/зачтено;

Особенности проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья обозначены в рабочей программе дисциплины (модуля).

Уровни сформированности компетенций определяется следующим образом:

1. Высокий уровень сформированности компетенций соответствует оценке отлично:

- предполагает формирование компетенций на высоком уровне;

- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки отлично;
- студент умеет применять на практике знания, полученные в рамках изучения дисциплины
- формируются навыки использования теоретических и практических разделов дисциплины для решения задач профессиональной деятельности;

2. Средний уровень соответствует оценке хорошо:

- предполагает формирование компетенций на среднем уровне;
- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки хорошо;
- студент умеет применять знания, полученные в рамках изучения дисциплины, для решения задач профессиональной деятельности;

3. Базовый уровень соответствует оценке удовлетворительно:

- предполагает формирование компетенций на базовом уровне;
- знание теоретических разделов изучаемой дисциплины на уровне не ниже оценки удовлетворительно;

4. Недостаточный уровень соответствует оценке неудовлетворительно.