МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ДГТУ)

ОЛИМПИАДА «Я – БАКАЛАВР» ДЛЯ ОБУЧАЮЩИХСЯ 5-11 КЛАССОВ

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПОДГОТОВКИ
К ОТБОРОЧНОМУ ЭТАПУ ОЛИМПИАДЫ
2025/2026 УЧЕБНОГО ГОДА ДЛЯ 10 КЛАССОВ

ОТБОРОЧНЫЙ ЭТАП

Отборочный этап олимпиады «Я – бакалавр» для обучающихся 10 классов (далее – Олимпиада) по предмету «Информатика» проходит дистанционно.

Вопросы заданий компонуются для каждого участника индивидуально в автоматическом режиме. Каждый вариант олимпиадной работы отборочного этапа включает в себя задания, предполагающие подготовленность участников Олимпиады в рамках ФГОС.

На решение задач отборочного этапа Олимпиады отводится 1 (один) астрономический час (60 минут). Отсчет времени начинается с момента начала выполнения заданий. Место и время выполнения заданий определяются участниками самостоятельно. Для выполнения заданий необходим компьютер с доступом в сеть Интернет. Оргкомитет не несет ответственности за сбои электропитания и связи в момент решения задач отборочного тура.

Участник Олимпиады может выполнять задания отборочного этапа однократно. В задания отборочного этапа входят 8 блоков вопросов. За каждый правильный ответ 1 блока участник получает 2 балла; за каждый правильный ответ 2 блока – 2 балла и так со всеми блоками. Максимально возможное количество набранных участником баллов – 100.

В олимпиадные задания отборочного тура включены элементы содержания из следующих разделов (тем) курса Информатика:

- раздел «Аппаратное обеспечение компьютера»;
- раздел «Программное обеспечение компьютера»;
- раздел «Позиционные системы счисления»;
- раздел «Логические основы компьютера»;
- раздел «Кодирование графической и звуковой информации, скорость передачи информации»;
- раздел «Программирование»;
- раздел «Работа в офисных программах»;
- раздел «Теоретические знания по информатике».

Для конструирования вариантов олимпиадной работы отборочного этапа использованы различные способы представления информации в текстах заданий (графики, таблицы, схемы и схематические рисунки).

Первый блок содержит задания на проверку теоретических знаний по аппаратной части персонального компьютера.

Второй блок содержит задания на проверку знаний по программной части персонального компьютера.

Третий блок содержит практические задания по правилам перевода из одной системы счисления в несколько других и обратно.

Четвёртый блок содержит задания на проверку знаний по логическим основам персонального компьютера.

Пятый блок содержит теоретические вопросы в виде тестов и практические задания для подсчёта объёма графического файла или звукового

файла, практические и теоретические вопросы по скорости передачи информации.

Шестой блок содержит практические задания и теоретические вопросы по программированию и алгоритмике.

Седьмой блок содержит практические вопросы по организации интерфейсной части персонального компьютера.

Восьмой блок содержит теоретические вопросы об основных понятиях информатики.

Участник Олимпиады получает индивидуальный вариант олимпиадной работы отборочного этапа, состоящий из вопросов: 6 заданий из первого блока заданий, 6 заданий из второго блока, 6 заданий из третьего блока, 6 заданий из четвертого блока, 6 заданий из пятого блока, 8 заданий из шестого блока, 6 заданий из седьмого блока, 6 заданий из восьмого блока.

Каждое задание оценивается в зависимости от уровня сложности и правильности полученного результата. Баллы, полученные участником Олимпиады за выполненные задания, суммируются.

ПЕРЕЧЕНЬ ЭЛЕМЕНТОВ СОДЕРЖАНИЯ, ВКЛЮЧЕННЫХ В ЗАДАНИЯ ОЛИМПИАДЫ ОТБОРОЧНОГО ЭТАПА 2025/2026 УЧЕБНОГО ГОДА

Блок 1. Аппаратное обеспечение компьютера.

Микропроцессор (МП). Это центральный блок ПК, предназначенный для управления работой всех блоков машины и для выполнения арифметических и логических операций над информацией.

Сопроцессор – специализированный процессор, предназначенный для совместной работы с некоторыми типами процессоров. Физически может находиться как внутри многих моделей МП, так и в виде дополнительной микросхемы на некоторых моделях системных плат. Сопроцессор дополняет возможности центрального процессора и расширяет набор команд (выполняет команды, не входящие в стандартный набор)**.** Он используется для ускоренного выполнения операций над двоичными числами с плавающей запятой, над двоично-кодированными десятичными числами, для вычисления некоторых трансцендентных (в том числе тригонометрических) функций.

Генератор тактовых импульсов генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет тактовую частоту машины. Промежуток времени между соседними импульсами определяет время одного такта работы машины или просто такт работы машины.

Частота генератора тактовых импульсов является одной из основных характеристик персонального компьютера и во многом определяет скорость его работы, потому что каждая операция в ПК выполняется за определенное количество тактов. В процессоре используется внутреннее умножение частоты, поэтому частота процессора в несколько раз больше, чем частота

системной шины. В современных компьютерах частота процессора может превышать частоту системной шины в сотни раз.

Системная шина – это основная интерфейсная (обеспечивающая взаимодействие) система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой. МП находящийся на системной плате должен взаимодействовать с периферийными устройствами. Фирмы DEC, IBM, а вслед за ними и другие фирмы выбрали так называемую открытую архитектуру. В рамках данной архитектуры устанавливается общий набор проводов, к которым подключены абсолютно идентичные разъемы или слоты.

Системная шина обеспечивает три направления передачи информации:

– между микропроцессором и основной памятью;

– между микропроцессором и портами ввода/вывода внешних устройств;

– между основной памятью и портами ввода/вывода внешних устройств (в режиме прямого доступа к памяти.)

Преимущества соединения через шину*:*

1. Шина позволяет подключать к ПЭВМ любые периферийные устройства, даже те устройства, которые будут созданы в будущем. Для этого достаточно только сделать соответствующий адаптер к периферийному устройству и вставить его в слот на системной плате. При закрытой архитектуре это невозможно, так как там идут только разводки к определенным устройствам, и только эти устройства могут быть подключены. Причем на стадии создания компьютера надо знать все о подключаемых периферийных устройствах, чтобы правильно сделать необходимые разводки.

Благодаря использованию фирмой IBM открытой архитектуры для компьютеров возникло множество фирм выпускающих адаптеры и различные периферийные устройства. Все это сделало компьютеры IBM и их клоны мировым стандартом.

2. Шина позволяет упростить и удешевить компьютеры, так как в этом случае на системной плате нет адаптеров периферийных устройств. Недостатком является тот факт, что в любой определенный момент времени МП может взаимодействовать только с одним периферийным устройством.

Итак, системная шина физически представляет собой систему проводников, с помощью которых системная плата соединяется с периферийными устройствами.

Количество параллельных проводников, по которым одновременно передаются сигналы, называется шириной шины.

Основная (или внутренняя) память предназначена для хранения и оперативного обмена информацией с прочими блоками машины. Внутренняя память содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ).

ПЗУ служит для хранения неизменной постоянной программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию (информацию в ПЗУ изменить нельзя).

ОЗУ предназначено для оперативной записи, хранения и считывания информации (программ и данных), непосредственно участвующей в информационно-вычислительном процессе, выполняемом ПК в текущий период времени. Главным достоинством оперативной памяти являются высокое быстродействие и возможность обращения к каждой ячейке памяти отдельно (прямой адресный доступ к ячейке). В качестве недостатка ОЗУ следует отметить невозможность сохранения информации в ней после выключения питания машины (энергозависимость).

Дополнительная (или внешняя) память относится к внешним устройствам ПК и используется для долговременного хранения любой информации, которая может когда-либо потребоваться для решения задач. В частности, во внешней памяти хранится все программное обеспечение компьютера. Внешняя память содержит разнообразные виды запоминающих устройств, но наиболее распространенными, имеющимися практически на любом компьютере, являются накопители на жестких (НЖМД) и гибких (НГМД) магнитных дисках.

Назначение этих накопителей – хранение больших объемов информации, запись и выдача хранимой информации по запросу в оперативное запоминающее устройство. Различаются НЖМД и НГМД лишь конструктивно, объемами хранимой информации и временем поиска, записи и считывания информации.

В качестве устройств внешней памяти используются также запоминающие устройства на кассетной магнитной ленте (стримеры), накопители на оптических дисках (НОД) и др.

Таймер. Это внутримашинные электронные часы, обеспечивающие автоматическое определение текущего момента времени (год, месяц, часы, минуты, секунды и доли секунд). Таймер подключается к автономному источнику питания (аккумулятору, батарейке на системной плате) и продолжает работать при отключении машины от сети.

Внешние устройства (ВУ). Это важнейшая составная часть любого вычислительного комплекса. Достаточно сказать, что по стоимости внешние устройства иногда составляют 50–80 % всего ПК. От состава и характеристик внешних устройств во многом зависят возможности и эффективность применения ПК в системах управления и в народном хозяйстве в целом.

ВУ ПК обеспечивают взаимодействие машины с окружающей средой: пользователями, объектами управления и другими ЭВМ. ВУ весьма разнообразны и могут быть классифицированы по ряду признаков. Так, по назначению можно выделить следующие виды ВУ:

– внешние запоминающие устройства (ВЗУ) или внешняя память;

– диалоговые средства пользователя;

– устройства ввода информации;

– устройства вывода информации;

– средства связи и коммуникации.

Диалоговые средства пользователя включают в свой состав видеомониторы (дисплеи), устройства речевого ввода-вывода информации.

К устройствам ввода информации относятся клавиатура, графические планшеты, сканеры, манипуляторы, сенсорные экраны.

К устройствам вывода информации относятся принтеры, графопостроители (плоттеры).

К устройствам связи и телекоммуникации относятся согласователи интерфейсов, адаптеры, цифро-аналоговые и аналого-цифровые преобразователи, сетевые интерфейсные платы, «стыки», мультиплексоры передачи данных, модемы и др.

Для согласования интерфейсов периферийные устройства подключаются к шине не напрямую, а через свои *контроллеры* (адаптеры) и *порты* примерно по такой схеме в соответствии с рисунком 1.

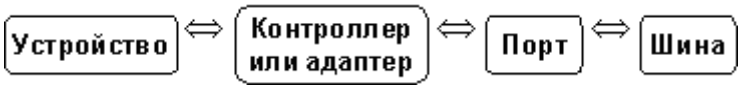


Рисунок 1 – Схема соединения внешних устройств

Адаптером можно назвать любое устройство, выполняющее функции согласования электронных, физических и других параметров для обеспечения взаимодействия устройств.

Контроллеры и адаптеры представляют собой наборы электронных цепей, которыми снабжаются устройства компьютера с целью совместимости их интерфейсов и обеспечения их связи с материнской платой. Контроллеры, кроме этого, осуществляют непосредственное управление периферийными устройствами по запросам микропроцессора. *Контроллер*, по сути, является адаптером со специализированным процессором, управляющим работой внешнего устройства по встроенной программе. Организация взаимодействия между внешним устройством и процессором осуществляется через аппаратный или программный порт ввода-вывода – канал передачи данных, представляемый как один или несколько адресов памяти, из которых можно прочитать или в которые можно записать данные.

Порты устройств представляют собой некие электронные схемы, содержащие один или несколько регистров ввода-вывода и позволяющие подключать периферийные устройства компьютера к внешним шинам микропроцессора.

Портами также называют *устройства стандартного интерфейса*: последовательный, параллельный и игровой порты (или интерфейсы).

К последовательным портам обычно подключаются двунаправленные устройства, которые должны как передавать информацию в компьютер, так и принимать ее.

Параллельные порты, как правило, используются для подключения принтеров и работают в однонаправленном режиме, хотя могут применяться и как двунаправленные.

Последовательный порт обменивается данными с процессором побайтно, а с внешними устройствами – побитно. Причем передача идет по одному проводку. *Параллельный порт* получает и посылает данные побайтно. В параллельных портах для передачи информации используется восемь линий. Этот интерфейс отличается высоким быстродействием. Существенным недостатком параллельного порта считается то, что соединительные провода не могут быть слишком длинными.

К *последовательному* порту обычно подсоединяют медленно действующие или достаточно удалённые устройства, такие, как мышь и модем. К *параллельному* порту подсоединяют более «быстрые» устройства – принтер и сканер. Через *игровой* порт подсоединяется джойстик. Клавиатура и монитор подключаются к своим *специализированным* портам через соответствующие *разъёмы*.

Основные электронные компоненты, определяющие архитектуру процессора, размещаются на основной плате компьютера, которая называется *системной* или *материнской* (MotherBoard). А контроллеры и адаптеры дополнительных устройств, либо сами эти устройства, могут выполняться в виде *плат расширения* (DaughterBoard – дочерняя плата) и подключаются к шине с помощью *разъёмов расширения***,** называемых также *слотами расширения* (англ. slot – щель, паз).

В настоящее время для настольных и портативных компьютеров разработана и широко применяется высокоскоростная универсальная последовательная шина получившая название USB (Universal Serial Bus). В шине USB реализована возможность подключения большого количества периферийных устройств к компьютеру. При подключении периферийного оборудования к персональному компьютеру, оснащенного шиной USB, его настройка происходит автоматически, сразу после физического подключения, без перезагрузки и установки.

Универсальная последовательная шина – это интерфейс, работающий со скоростью до 480 Мбит/сек (в спецификации USB 2.0) и основанный на простом 4-проводном соединении. Эта шина поддерживает до 127 подключаемых USB устройств и использует топологию звезды, расширяемой концентраторами, которые могут входить как в персональный компьютер, так и в подключаемые устройства в соответствии с рисунком 1.

Рисунок 1 – Основные периферийные устройства

Примеры заданий:

Вопрос 1. Какие устройства относятся к устройствам ввода информации?

- ☐ Клавиатура
- ☐ Монитор
- ☐ Сканер
- ☐ Принтер
- ☐ Цифровая камера.

Ответ: Клавиатура, Сканер, Цифровая амера

Вопрос 2. Виды мониторов:

- ☐ Матричный
- ☐ Жидкокристаллический
- ☐ на электронно-лучевой трубке
- ☐ Лазерный

Ответ: Жидкокристаллический, На электронно-лучевой трубке

Вопрос 3. Устройство, предназначенное для вывода сложных и широкоформатных графических объектов

- ○ Принтер
- ○ Колонки
- ○ Проектор
- ○ Плоттер

Ответ: Плоттер

Раздел 2. Программное обеспечение компьютера.

В основу работы любого компьютера положен программный принцип управления, состоящий в том, что компьютер выполняет действия по заранее заданной программе.

Программа – это запись алгоритма решения задачи в виде последовательности команд или операторов на языке, который понимает компьютер.

Конечная цель любой компьютерной программы – управление аппаратными средствами. Даже если на первый взгляд программа не взаимодействует с оборудованием, не требует никакого ввода данных с устройств ввода и не осуществляет вывод данных на устройства вывода, все равно ее работа основана на управлении аппаратными устройствами компьютера.

Работа компьютерной системы осуществляется в непрерывном взаимодействии аппаратных и программных средств.

Программное обеспечение (ПО, Software) – это совокупность программ и соответствующей документации, позволяющая использовать вычислительную технику для решения различных задач.

Программное обеспечение выполняет следующие основные функции:

– обеспечивает работоспособность ЭВМ, так как без соответствующего ПО компьютеры не могут осуществлять никакие операции;

– расширяет ресурсы вычислительной системы и повышает эффективность их использования;

– облегчает взаимодействие пользователя с ЭВМ и повышает производительность его труда, т. е. обеспечивает пользовательский интерфейс.

Состав программного обеспечения вычислительной системы называют программной конфигурацией. Между программами, как и между физическими узлами и блоками, существует взаимосвязь – многие программы работают, опираясь на другие программы более низкого уровня, то есть мы можем говорить о программном интерфейсе. Программный интерфейс — функциональность, которую некоторый программный компонент предоставляет другим программным компонентам. Возможность существования такого интерфейса тоже основана на существовании технических условий и протоколов взаимодействия. На практике он обеспечивается распределением программного обеспечения на несколько взаимодействующих между собой уровней. Уровни программного обеспечения можно представить в виде пирамидальной конструкции, каждый вышестоящий уровень которой опирается на программное обеспечение предшествующих уровней, а сам, в свою очередь, повышает функциональность всей системы.

Примеры заданий:

Вопрос 1. Операционная система - это

- система математических операций для решения отдельных задач
- система программ, которая обеспечивает совместную работу всех устройств компьютера по обработке информации
- набор программ для работы устройства системного блока компьютера

Ответ: система программ, которая обеспечивает совместную работу всех устройств компьютера по обработке информации

Вопрос 2. Программное обеспечение (ПО) - это:

- совокупность программ, позволяющих организовать решение задач на компьютере
- список имеющихся в кабинете программ, заверенных администрацией школы
- программы для организации совместной работы устройств компьютера как единой системы

Ответ: совокупность программ, позволяющих организовать решение задач на компьютере

Вопрос 3. Системное программное обеспечение - это

- программы для организации удобной системы размещения программ на диске
- программы для организации совместной работы устройств компьютера как единой системы
- набор программ для работы устройства системного блока компьютера

Ответ: программы для организации удобной системы размещения программ на диске

Раздел 3. Позиционные системы счисления.

Под **системой счисления** понимается способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами. Все системы счисления делятся на **позиционные** и **непозиционные**.

Непозиционными называются такие системы счисления, в которых каждый символ сохраняет свое значение независимо от места его положения в числе. Примером непозиционной системы счисления является римская система, в которой символам I, V, X, L, C, D, M соответствуют числа 1, 5, 10, 50, 100, 500, 1000. Недостатком этой системы является сложность формальных правил записи чисел и выполнения арифметических действий над ними.

Система счисления называется **позиционной**, если значение каждого знака в числе зависит от позиции, которую занимает знак в записи числа. Это значение находится в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. Примером позиционной системы счисления является десятичная система, используемая в повседневной жизни.

Количество различных цифр, употребляемых в позиционной системе, определяет название системы счисления и называется **основанием** системы счисления. Так, в десятичной системе используются десять цифр (от 0 до 9), основанием этой системы является число десять.

В позиционных системах счисления числа записываются в виде последовательности символов:

$$N = a_n\, a_{n-1} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-m\,(p)} \qquad (1)$$

где N – число;

a_i – цифры (символы) числа;

p – основание системы счисления;

n, m – порядковый номер разряда для целой (n) и дробной (m) частей числа соответственно.

В этой последовательности запятая отделяет целую часть числа от дробной (коэффициенты при положительных степенях, включая нуль, от коэффициентов при отрицательных степенях). Значение числа, записанного в виде (1), может быть найдено по следующей формуле:

$$N = a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + \ldots + a_0 \cdot p^0 + a_{-1} \cdot p^{-1} + a_{-2} \cdot p^{-2} + \ldots + a_{-m} \cdot p^{-m} \,. \qquad (2)$$

В десятичной системе счисления мы производим вычисления по формуле (2) практически не задумываясь. Возьмём для примера десятичное число 123,45:

$$\overset{2\ 1\ 0\ -1\ -2}{123{,}45}_{(10)} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2} = 100 + 20 + 3 + 0{,}4 + 0{,}05$$

Здесь и в дальнейшем основание системы счисления, в которой представлено число, будем указывать в виде нижнего индекса в скобках.

Помимо десятичной, в ЭВМ применяются и другие позиционные системы счисления: двоичная, восьмеричная, шестнадцатеричная.

Двоичная система счисления.

Используется две цифры: 0 и 1. Особая значимость двоичной системы счисления в информатике определяется тем, что внутреннее представление любой информации в компьютере является двоичным кодом. Примеры представления чисел в двоичной системе счисления представлены в таблице 1.

Восьмеричная система счисления.

Используется восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7. Употреблялась в ЭВМ первого и второго поколений как вспомогательная для записи адресов и

данных в сокращенном виде. Для представления одной цифры восьмеричной системы используется три двоичных разряда (триада) (Таблица 1). Триада получается путем добавления, при необходимости, незначащих нулей.

Шестнадцатеричная система счисления.

Для изображения чисел употребляются 16 цифр. Первые десять цифр этой системы обозначаются цифрами от 0 до 9, а старшие шесть цифр - латинскими буквами: 10-A, 11-B, 12-C, 13-D, 14-E, 15-F. Шестнадцатеричная система используется для записи информации в сокращенном виде. Для представления одной цифры шестнадцатеричной системы счисления используется четыре двоичных разряда (тетрада, или полубайт) (Таблица 1).

Таблица 1. – Представление чисел в различных системах счисления

Десятичная (Основание 10)	Римская	Двоичная (основание 2)	Восьмеричная (Основание 8)	Двоичная (триады)	Шестнадцатеричная (Основание 16)	Двоичная (тетрады)
		0	0	000	0	0000
	I	1	1	001	1	0001
	II	10	2	010	2	0010
	III	11	3	011	3	0011
	IV	100	4	100	4	0100
	V	101	5	101	5	0101
	VI	110	6	110	6	0110
	VII	111	7	111	7	0111
	VIII	1000	10	001 000	8	1000
	IX	1001	11	001 001	9	1001
0	X	1010	12	001 010	A	1010
1	XI	1011	13	001 011	B	1011
2	XII	1100	14	001 100	C	1100
3	XIII	1101	15	001 101	D	1101
4	XIV	1110	16	001 110	E	1110
5	XV	1111	17	001 111	F	1111
6	XVI	10000	20	010 000	10	0001 0000
7	XVII	10001	21	010 001	11	0001 0001

Перевод чисел в десятичную систему осуществляется путем составления степенного ряда (2) с основанием той системы, из которой число переводится. Затем подсчитывается значение суммы.

Перевод целых десятичных чисел в недесятичную систему счисления осуществляется последовательным делением десятичного числа на основание той системы, в которую оно переводится, до тех пор, пока не

получится частное, меньшее этого основания. Число в новой системе записывается в виде остатков деления, начиная с последнего.

Перевод правильных дробей из десятичной системы счисления в недесятичную. Для перевода правильной десятичной дроби в другую систему эту дробь надо последовательно умножать на основание той системы, в которую она переводится. При этом умножаются только дробные части. Дробь в новой системе записывается в виде целых частей произведений, начиная с первого.

Замечание. Конечной десятичной дроби может соответствовать бесконечная (периодическая) дробь в недесятичной системе счисления. В этом случае количество знаков в представлении дроби в новой системе берется в зависимости от требуемой точности.

Для перевода неправильной десятичной дроби в систему счисления с недесятичным основанием необходимо отдельно перевести целую часть и отдельно дробную.

Необходимо отметить, что целые числа остаются целыми, а правильные дроби - дробями в любой системе счисления.

Для перевода восьмеричного или шестнадцатеричного числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трехразрядным двоичным числом (триадой) – для восьмеричной системы счисления или четырехразрядным двоичным числом (тетрадой) – для шестнадцатеричной системы счисления (таблица 1), после чего отбрасывают незначащие нули в старших и младших разрядах.

Для перехода от двоичной к восьмеричной (шестнадцатеричной) системе поступают следующим образом: двигаясь от десятичной точки влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем триаду (тетраду) заменяют соответствующей восьмеричной (шестнадцатеричной) цифрой (таблица 1).

Перевод из восьмеричной в шестнадцатеричную систему и обратно удобно осуществлять через двоичную систему с помощью триад и тетрад.

Примеры заданий:

Вопрос 1. Переведите число из двоичной системы счисления в десятичную.

Разбор задания.

а) Перевести $10101101{,}101_{(2)}$ в десятичную систему счисления

$10101101{,}101_{(2)} = 1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + + 0\cdot2^{-2} + 1\cdot2^{-3} = 173{,}625_{(10)}$

Ответ: 173,625 (10)

Вопрос 2. Переведите число из восьмеричной системы счисления в десятичную.

Разбор задания.

Перевести $703,04_{(8)}$ в десятичную систему счисления

$703,04_{(8)} = 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 + 0 \cdot 8^{-1} + 4 \cdot 8^{-2} = 451,0625_{(10)}$

Ответ: 451,0625

Вопрос 3. Переведите число из шестнадцатеричной системы счисления в десятичную.

Разбор задания.

Перевести $B2E,4_{(16)}$ в десятичную систему счисления

$B2E,4_{(16)} = 11 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 + 4 \cdot 16^{-1} = 2862,25_{(10)}$

Ответ: $2862,25_{(10)}$

Раздел 4. Логические основы компьютера

Слово «логика» означает как совокупность правил, которым подчиняется процесс мышления, так и науку о правилах рассуждений. Логика, как наука о законах и формах мышления, изучает абстрактное мышление как средство познания объективного мира.

Основными формами абстрактного мышления являются:

— ПОНЯТИЯ,

— СУЖДЕНИЯ,

— УМОЗАКЛЮЧЕНИЯ.

ПОНЯТИЕ — форма мышления, в которой отражаются существенные признаки отдельного предмета или класса однородных предметов, например: «портфель»; «трапеция»; «ветер».

СУЖДЕНИЕ — мысль, в которой что-либо утверждается или отрицается о предметах. Суждения являются истинными или ложными повествовательными предложениями. Они могут быть простыми и сложными. Например: «Весна наступила»; «Грачи прилетели»; «Весна наступила, и грачи прилетели».

УМОЗАКЛЮЧЕНИЕ — прием мышления, посредством которого из исходного знания получается новое знание; из одного или нескольких истинных суждений, называемых посылками, мы по определенным правилам вывода получаем заключение.

Все металлы — простые вещества.

Литий — металл.

Литий — простое вещество.

Чтобы достичь истины при помощи умозаключений, надо соблюдать законы логики. Существует формальная и математическая логика.

Формальная логика — наука о законах и формах мышления.

Математическая логика изучает логические связи и отношения, лежащие в основе дедуктивного (логического) вывода.

Формальная логика связана с анализом наших обычных содержательных умозаключений, выражаемых разговорным языком. Математическая логика изучает только умозаключения со строго определенными объектами и

суждениями, для которых можно однозначно решить, истинны они или ложны.

В основе логических схем и устройств ЭВМ лежит специальный аппарат, использующий законы математической логики. Математическая логика изучает вопросы применения математических методов для решения логических задач и построения логических схем. Знание логики необходимо при разработке алгоритмов и программ, так как в большинстве языков программирования есть логические операции.

Алгебра логики — это раздел математической логики, значения всех элементов (функций и аргументов) которой определены в двухэлементном множестве: «Истина» («True») и «Ложь» («False»), или 1 и 0.

В математической логике суждения называются высказываниями. Алгебру логики иначе называют алгеброй высказываний.

Высказывание — это повествовательное предложение, о котором можно сказать, истинно оно или ложно.

Примеры высказываний:

Сейчас идет снег. м*ожет быть истинным или ложным*

Земля — планета Солнечной системы. *истинно*

2 + 8 < 5 *ложно*

5 * 5 = 25*истинно*

Всякий квадрат есть параллелограмм. и*стинно*

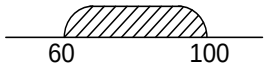
Всякий параллелограмм есть квадрат. *ложно*

2 * 2 = 5 *ложно*

А вот примеры, не являющиеся высказываниями: «Уходя, гасите свет!»; «Да здравствует мыло душистое и полотенце пушистое!»

Высказывания, приведенные выше, являются простыми. Сложные высказывания получаются путем объединения простых высказываний связками — союзами И, ИЛИ и частицей НЕ. Значение истинности сложных высказываний зависит от истинности входящих в них простых высказываний и от объединяющих их связок.

Примеры заданий:

Вопрос 1. Определить, что сумма баллов S**,** набранная студентом на тестировании находится в пределах 60÷100 баллов, то есть принадлежит интервалу [60,100] 

:

Ответ: S >= 60 AND S <= 100

Вопрос 2. Определить, что сумма баллов S**,** набранная студентом на тестировании не входит в пределы 60÷100 баллов, то есть находится вне интервала [60,100] 

:

Ответ: S >= 60 AND S <= 100

Вопрос 3. Представьте результат работы логической операции инверсия

Ответ:

X	Y	NOT X
1	1	0
1	0	0
0	1	1
0	0	1

Раздел 5. Кодирование графической и звуковой информации, скорость передачи информации

Кодирование и преобразование графической и звуковой информации. Определение скорости передачи информации при заданной пропускной способности канала

Формула объема памяти для хранения растрового изображения:

$$I = M * N * i$$

I — объем памяти, требуемый для хранения изображения, измеряется в битах

M — ширина изображения в пикселях

N — высота изображения в пикселях

i — глубина кодирования цвета или разрешение

Или можно формулу записать так:

$$I = N * i \text{ битов}$$

где N – количество пикселей (M * N) и i – глубина кодирования цвета (разрядность кодирования)

Следует также помнить формулы преобразования:

1 Мбайт = 2^{20} байт = 2^{23} бит,

1 Кбайт = 2^{10} байт = 2^{13} бит

Пиксель – это наименьший элемент растрового изображения, который имеет определенный цвет.

Разрешение – это количество пикселей на дюйм размера изображения.

Глубина цвета — это количество битов, необходимое для кодирования цвета пикселя.

Если глубина кодирования составляет i битов на пиксель, код каждого пикселя выбирается из 2^i возможных вариантов, поэтому можно использовать не более 2^i различных цветов.

Формула для нахождения количества цветов в используемой палитре:

$$i = \log_2 N$$

N — количество цветов

i — глубина цвета

Примеры заданий:

Вопрос 1. Музыкальный фрагмент был оцифрован и записан в виде файла без использования сжатия данных. Получившийся файл был передан в город А по каналу связи за 30 секунд. Затем тот же музыкальный фрагмент был оцифрован повторно с разрешением в 2 раза выше и частотой дискретизации в 1,5 раза меньше, чем в первый раз. Сжатие данных не производилось. Полученный файл был передан в город Б; пропускная способность канала связи с городом Б в 4 раза выше, чем канала связи с городом А. Сколько секунд длилась передача файла в город Б? В ответе запишите только целое число, единицу измерения писать не нужно.

Разбор задания: объём музыкального файла вычисляется по формуле $I = f \cdot r \cdot k \cdot t$, где f – частота дискретизации, r – разрешение (глубина кодирования), k – количество каналов, t – время звучания

при повышении разрешения (количества битов на хранения одного отсчёта) в 2 раза объём файла (при прочих равных условиях) увеличивается в 2 раза, поэтому время тоже увеличится в 2 раза

при снижении частоты дискретизации (количества хранимых отсчётов за 1 секунду) в 1,5 раза объём файла (при прочих равных условиях) уменьшается в 1,5 раза, поэтому время тоже уменьшится в 1,5 раза

при увеличении пропускной способности канала связи (здесь это то же самое, что и скорость передачи данных) в 4 раза время передачи (при прочих равных условиях) уменьшится в 4 раза

поэтому исходное время передачи файла нужно

а) умножить на 2

б) разделить на 1,5

в) разделить на 4

получается 30 · 2 / 1,5 / 4 = 10 секунд

Ответ: 10.

Вопрос 2. Производилась двухканальная (стерео) звукозапись с частотой дискретизации 64 кГц и 24-битным разрешением. В результате был получен файл размером 120 Мбайт, сжатие данных не производилось. Определите приблизительно, сколько времени (в минутах) производилась запись. В качестве ответа укажите ближайшее к времени записи целое число, кратное 5.

Разбор задания: так как частота дискретизации 64 кГц, за одну секунду запоминается 64000 значений сигнала так как глубина кодирования – 24 бита = 3 байта, для хранения 1 секунды записи требуется

2 × 64000 × 3 байта

(коэффициент 2 – для стерео записи)

на 1 минуту = 60 секунд записи потребуется

60 × 2 × 64000 × 3 байта

переходим к степеням двойки, заменяя 60 ← 64 = 26; 1000 ← 1024 = 210:

26 × 21 × 26 × 210 × 3 байта = 26 × 21 × 26 × 3 Кбайта

= 22 × 21 × 3 Мбайта = 24 Мбайта

тогда время записи файла объёмом 120 Мбайт равно 120 / 24 = 5 минут

Ответ: 5.

Вопрос 3. Для хранения в информационной системе документы сканируются с разрешением 200 dpi и цветовой системой, содержащей 130 цветов. Методы сжатия изображений не используются. Средний размер отсканированного документа составляет 10 Мбайт. Для повышения качества представления информации было решено перейти на разрешение 300 dpi и цветовую систему, содержащую 216 = 65 536 цветов. Сколько Мбайт будет составлять средний размер документа, отсканированного с изменёнными параметрами?

Разбор задания: Разрешение изображения изменилось с 200 dpi на 300 dpi. Это означает, что размер изображения изменился в (300 · 300) / (200 · 200) = 9/4 раз. Для хранения цветовой системы, состоящей из 130 цветов, необходимо 8 бит (необходимо подобрать минимально возможный i, так, чтобы $N \le 2i$; $130 \le 28$). Глубина кодирования изображения изменилась с 8 бит до 16 бит. Это означает, что размер изображения изменился в 16/8 = 2 раза. Средний размер документа, отсканированного с изменёнными параметрами равно 10 · 9/4 · 2 = 45 Мб.

Ответ: 45.

Раздел 6. Программирование

Программирование — это процесс и искусство создания компьютерных программ с помощью языков программирования. В более широком смысле оно представляет собой разработку программного обеспечения. Программирование сочетает в себе элементы искусства, науки, математики и инженерии и подразумевает написание исходного кода, который потом компилируется или интерпретируется в машинный код для выполнения на компьютере.

Основная задача программирования — это реализация алгоритмов, то есть последовательностей действий, которые компьютер должен выполнить для решения конкретной задачи. При этом программисты выбирают подходящий язык программирования, учитывая тип задачи и требования к производительности и удобству разработки.

Основные понятия в программировании включают переменные (хранение данных), инструкции (команды для компьютера) и выражения (вычисления). Программирование широко используется в самых разных сферах: от разработки сайтов и приложений до автоматизации бизнес-процессов, медицины и образования

Каждый язык имеет свои собственные особенности и тонкости реализации, которые можно узнать на официальных страницах этих языков.

Например:

Python https://www.python.org/

Java https://www.java.com/ru/

C# https://dotnet.microsoft.com/ru-ru/languages/csharp
C++ https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170

Эти ссылки являются основными для получения теоретических знаний о построении и особенностях применения конкретного языка программирования.

Для подготовки к тестированию по блоку программирование необходимо изучить семь основных блоков:

1. Общие вопросы программирования.

Для успешного освоения программирования необходимо изучить широкий спектр общих вопросов. Начните с основ. Изучите, что такое алгоритм, как его представить в виде блок-схемы или псевдокода, и освойте основные алгоритмические структуры, такие как последовательность, ветвление и циклы, а также базовые алгоритмы поиска и сортировки.

Углубитесь в понимание типов данных, включая основные (целые числа, числа с плавающей точкой, символы, строки, логические значения) и структуры данных (массивы, связные списки, стеки, очереди, деревья, графы, хэш-таблицы). Необходимо понимать, как объявлять и инициализировать переменные и константы, а также учитывать область их видимости. Освойте арифметические, логические операторы, операторы сравнения и присваивания.

Изучите парадигмы программирования, такие как императивное, декларативное, структурное, объектно-ориентированное и функциональное программирование, а также принципы SOLID. Необходимо понимать структуры управления программой, включая условные операторы, циклы, функции, процедуры и обработку исключений.

Необходимо ориентироваться в терминах, что такое алгоритм, компилятор, интерпретатор, синтаксис, исходный код, парадигма программирования, отладка программы, API, IDE, и т.д. Знать основы работы с программным кодом.

2. Типы и структуры данных.

В программировании типы и структуры данных играют ключевую роль в организации и эффективной обработке информации. Фундаментом являются примитивные типы данных, такие как целые числа (int, short, long, byte), числа с плавающей точкой (float, double), символы (char), строки (String) и логические значения (boolean). Важно понимать их особенности и диапазоны значений.

Составные типы, такие как массивы и структуры, строятся на основе примитивных типов, позволяя группировать связанные данные.

Структуры данных, в свою очередь, предоставляют различные способы организации и хранения, влияя на производительность программы.

Линейные структуры включают связные списки, стеки, очереди и деки, каждая из которых имеет свои особенности и принципы работы (LIFO, FIFO).

Нелинейные структуры, такие как деревья и графы, предоставляют иерархические и сетевые способы организации данных.

Ассоциативные структуры, такие как хэш-таблицы, словари и множества, обеспечивают быстрый поиск и хранение уникальных элементов.

Абстрактные типы данных (ADT) определяют что структура данных делает, а не как. Список, стек, очередь, дерево и граф являются примерами ADT.

Выбор подходящей структуры данных зависит от требований к производительности, объема данных, типов операций и простоты реализации.

При изучении необходимо понимать концепции указателей и ссылок, динамического выделения памяти, рекурсии, обобщенного программирования и сложности алгоритмов. Рекомендуется изучать теорию, практиковаться в реализации структур данных, рассматривать примеры их использования и анализировать эффективность различных структур для конкретных задач**.**

3. Условные конструкции.

Разберитесь с базовым синтаксисом работы условных конструкций. Для этого рассмотрите операторы:

if (если): Самый простой условный оператор. Выполняет блок кода только в том случае, если условие истинно.

```
if условие:
    # Код, который выполняется, если условие истинно
```

else (иначе): Выполняет блок кода, если условие if ложно.

```
if условие:
    # Код, который выполняется, если условие истинно
else:
    # Код, который выполняется, если условие ложно
```

elif (иначе если) / else if: Позволяет проверять несколько условий последовательно. Выполняется блок кода, соответствующий первому истинному условию.

```
if условие1:
    # Код, если условие1 истинно
elif условие2:
    # Код, если условие2 истинно
else:
```

```
    # Код, если ни одно из условий не истинно
```

Условия (логические выражения).
Операторы сравнения:
== (равно)
!= (не равно)
> (больше)
< (меньше)
>= (больше или равно)
<= (меньше или равно)

Логические операторы:
and / && (логическое И). Условие истинно, если оба операнда истинны.
or / || (логическое ИЛИ). Условие истинно, если хотя бы один операнд истинен.
not / ! (логическое НЕ). Инвертирует значение операнда (истина становится ложью, и наоборот).

Оператор **switch** (в некоторых языках):
Альтернатива цепочке **if-elif-else** для проверки равенства переменной с несколькими конкретными значениями.
Пример для c++

```
switch (переменная) {
   case значение1:
      // Код для значения1
      break;
   case значение2:
      // Код для значения2
      break;
   default:
      // Код, если ни одно из значений не совпало
}
```

Важно помнить про break, чтобы избежать "проваливания" в следующие case.

Пример тестового задания.
1. Что такое "ветка else" в условной конструкции if-else?
a) Блок кода, который выполняется, если условие в операторе if ложно.
b) Блок кода, который выполняется, если условие в операторе if истинно.
c) Блок кода, который всегда выполняется.
d) Блок кода, который выполняется, если произошла ошибка.

Разбор задания:

a) Блок кода, который выполняется, если условие в операторе if ложно. Это правильное описание. Ветка else предназначена для выполнения кода в случае, когда условие в if не выполняется (т.е., ложно).

b) Блок кода, который выполняется, если условие в операторе if истинно. Это неверно. Блок кода, который выполняется при истинном условии, находится внутри блока if.

c) Блок кода, который всегда выполняется. Это неверно. Ветка else выполняется только тогда, когда условие в if ложно.

d) Блок кода, который выполняется, если произошла ошибка. Это неверно. Обработка ошибок обычно выполняется с помощью других механизмов, таких как try-except (в Python) или try-catch (в Java, C++). Хотя в блоке else можно предусмотреть обработку определенных ситуаций, которые могли возникнуть при ложном условии, его основное назначение - не обработка ошибок.

Вывод. Исходя из вопроса и предложенных ответов, правильным ответом является - a) Блок кода, который выполняется, если условие в операторе if ложно.

4. Массивы.

Массив – это упорядоченная структура данных, содержащая элементы одного типа, которые расположены в памяти последовательно. Каждый элемент доступен по своему индексу, начиная, как правило, с нуля.

Массивы бывают статические, размер которых фиксируется во время компиляции и не может быть изменен, и динамические, размер которых можно изменять во время выполнения программы. Также существуют многомерные массивы, представляющие собой массивы массивов и используемые для представления, например, таблиц.

Основные операции над массивами включают создание, доступ к элементам по индексу (чтение и запись), итерацию (перебор всех элементов), изменение размера (для динамических массивов), поиск элементов и сортировку. Важно помнить об ошибке выхода за границы массива.

Существуют различные алгоритмы для работы с массивами, такие как линейный и бинарный поиск, а также алгоритмы сортировки: пузырьковая сортировка, сортировка вставками, выбором, быстрая сортировка и сортировка слиянием. Важно понимать, как сложность алгоритмов (оценка их эффективности) влияет на время выполнения операций с массивами, особенно с большими объемами данных.

В разных языках программирования массивы имеют свои особенности. В C/C++ важен контроль типов и ручное управление памятью (или использование умных указателей), в Java массивы являются объектами, а в

Python списки предоставляют удобные встроенные методы для работы с данными.

В Python и JavaScript поддерживаются динамические массивы с гибкой типизацией и автоматическим управлением памятью.

Для изучения массивов рекомендуется начать с теории, затем перейти к практике решения задач, использовать отладчик для понимания работы кода и анализировать существующие примеры кода.

Пример тестового задания.

Что произойдет, если обратиться к элементу массива по индексу, находящемуся за пределами допустимого диапазона?

a) Программа продолжит выполнение, элемент массива будет создан автоматически.

b) Программа выдаст предупреждение, но продолжит выполнение.

c) Произойдет ошибка.

d) Программа закроется.

Разбор.

a) Программа продолжит выполнение, элемент массива будет создан автоматически. Это неверно. Массивы имеют фиксированный размер (или, если это динамический массив, то доступ за его текущий размер также недопустим), и добавление элементов "автоматически" при обращении по недопустимому индексу не происходит.

b) Программа выдаст предупреждение, но продолжит выполнение. В некоторых редких случаях (в зависимости от языка программирования и настроек компилятора) может быть выдано предупреждение, но это не является обычной ситуацией. Чаще всего, это приведет к более серьезным последствиям.

c) Произойдет ошибка. Это наиболее общий и правильный ответ. В большинстве языков программирования (C++, Java, Python и др.) обращение к элементу массива по индексу за пределами допустимого диапазона приведет к возникновению ошибки времени выполнения (например, ArrayIndexOutOfBoundsException в Java или segmentation fault в C/C++).

d) Программа закроется. В некоторых случаях ошибка выхода за границы массива может привести к критическому сбою, в результате которого программа аварийно завершится. Однако в других случаях (особенно если ошибка обрабатывается) программа может не закрыться целиком, но ее дальнейшее поведение может быть непредсказуемым. Вариант (c) более точно описывает универсальный сценарий.

Вывод. Исходя из вопроса и предложенных ответов, правильным ответом является - c) Произойдет ошибка.

5. Циклы.

Цикл в программировании – это конструкция, позволяющая многократно выполнять определенный блок кода, который называется телом цикла. Повторение происходит до тех пор, пока выполняется заданное условие цикла.

В циклах часто используется счетчик (итератор) – переменная, которая изменяется на каждой итерации и помогает контролировать выполнение цикла.

Существуют три основных типа циклов:

— for (цикл со счетчиком);

— while (цикл "пока");

— do-while (цикл "делай-пока").

Синтаксис циклов немного отличается в разных языках программирования, но основная логика остается одинаковой.

Управление циклом осуществляется с помощью операторов break (немедленный выход из цикла) и continue (пропуск текущей итерации). Важно избегать создания бесконечных циклов, которые могут возникнуть, если неверно задано условие выхода.

Вложенные циклы – это циклы, помещенные внутри других циклов, которые полезны для обработки многомерных массивов и других сложных структур данных.

Циклы находят широкое применение в программировании, например, для обработки массивов, чтения данных из файлов, создания графических интерфейсов и реализации различных алгоритмов. Важно уметь оптимизировать циклы, уменьшая количество итераций и вынося инвариантный код за их пределы.

При работе с циклами часто возникают ошибки, такие как бесконечные циклы, смещение индекса, неправильная инициализация счетчика и неверное использование break и continue.

Использование отладчика и анализ существующего кода помогают избежать этих ошибок и лучше понять работу циклов. В некоторых языках существуют продвинутые концепции, такие как итераторы и генераторы в Python, которые предоставляют более эффективные способы перебора последовательностей. Также бывают доступны циклы foreach и параллельные циклы для упрощения работы с коллекциями и повышения производительности.

Пример тестового задания.

Что произойдет, если условие в цикле while всегда истинно?

a) Цикл выполнится только один раз.

b) Программа выдаст ошибку.

c) Цикл будет выполняться бесконечно.

d) Ничего, программа продолжит работу, не выполняя цикл.

Разбор.

a) Цикл выполнится только один раз. Это неверно. Цикл while продолжает выполняться, пока условие истинно. Если условие всегда истинно, то цикл продолжит выполняться снова и снова.

b) Программа выдаст ошибку. Это неверно. Сам по себе бесконечный цикл не вызывает ошибку времени выполнения (если только внутри цикла не происходит что-то, что приводит к ошибке, например, переполнение памяти).

c) Цикл будет выполняться бесконечно. Это правильный ответ. Если условие в цикле while всегда истинно, то цикл будет повторяться без остановки, пока не будет принудительно прерван (например, операционной системой или если внутри цикла есть оператор break, который никогда не выполняется). Такой цикл называется бесконечным циклом.

d) Ничего, программа продолжит работу, не выполняя цикл. Это неверно. Цикл while должен начать выполняться, так как условие изначально истинно.

Вывод. Исходя из вопроса и предложенных ответов, правильным ответом является - c) Цикл будет выполняться бесконечно.

6. Функции.

Функция в программировании – это организованный, многократно используемый блок кода, предназначенный для выполнения определенной задачи.

Использование функций повышает модульность программы, позволяя разбивать сложный код на более мелкие, понятные части, а также обеспечивает повторное использование кода, избегая дублирования. Кроме того, они улучшают читаемость и упрощают отладку и тестирование.

Функция состоит из имени, параметров (аргументов), возвращаемого значения и тела функции. Параметры могут быть позиционными, именованными или иметь значения по умолчанию.

Существуют разные типы параметров, такие как формальные (в определении функции) и фактические (при вызове). Также функции могут принимать переменное число аргументов.

Важно понимать область видимости переменных. Локальные переменные видны только внутри функции, а глобальные – во всей программе, хотя их использование следует ограничивать. Параметры могут передаваться по значению (копируется значение аргумента), по ссылке (передаётся ссылка на переменную) или по указателю (передаётся адрес памяти).

Функции могут возвращать значение или не возвращать его (процедуры). Рекурсия – это когда функция вызывает саму себя, требующая базового случая для прекращения вызовов, иначе возникнет бесконечный цикл. Функции высшего порядка принимают другие функции в качестве

аргументов или возвращают их. Лямбда-функции — это короткие анонимные функции.

При работе с функциями важно придерживаться общих походов, таких как понятные имена, небольшой размер функций, документирование и тестирование.

Необходимо избегать побочных эффектов, чтобы функция не меняла глобальное состояние программы. В языках программирования существуют продвинутые концепции, например, замыкания, декораторы и шаблоны функций, а также практики функционального программирования.

Пример тестового задания.
Что такое "параметр" функции?
a) Результат, который функция возвращает после выполнения.
b) Имя функции.
c) Значение, передаваемое в функцию при ее вызове.
d) Локальная переменная внутри функции.

Разбор.
a) Результат, который функция возвращает после выполнения. Это неверно. Результат, который функция возвращает, называется возвращаемым значением (return value), а не параметром.
b) Имя функции. Это неверно. Параметр – это не имя функции, а аргумент, который передается в функцию.
c) Значение, передаваемое в функцию при ее вызове. Это правильный ответ. Параметры – это значения, которые передаются в функцию для использования внутри нее. Эти значения позволяют функции работать с разными данными и выполнять разные действия в зависимости от входных данных.
d) Локальная переменная внутри функции. Это неверно. Локальные переменные – это переменные, объявленные внутри функции и используемые для хранения промежуточных результатов или других данных, необходимых для выполнения функции. Они не передаются извне, а существуют только внутри функции.

Вывод. Исходя из вопроса и предложенных ответов, правильным ответом является - c) Значение, передаваемое в функцию при ее вызове.

7. Объектно-ориентированное программирование.

Объектно-ориентированное программирование (ООП) — это метод программирования, где данные представляются в виде объектов, сочетающих в себе информацию (атрибуты) и действия (методы). Вместо акцента на последовательности шагов, как в процедурном подходе, ООП организует код вокруг этих объектов, имитируя реальные вещи.

Ключевые идеи ООП включают классы и объекты, где класс — это шаблон, а объект — конкретный пример.

Инкапсуляция скрывает внутреннее устройство объекта, предоставляя только нужный интерфейс для работы с ним.

Наследование позволяет создавать новые классы на основе старых, перенимая их свойства.

Полиморфизм дает возможность работать с разными типами объектов единообразным способом.

Абстракция упрощает сложные системы, показывая только главное и скрывая детали.

Преимущества ООП в том, что оно делает код более модульным, позволяя разбивать большие задачи на части, которые легче разрабатывать и поддерживать. Оно позволяет повторно использовать код через наследование, делая разработку быстрее и уменьшая количество ошибок. ООП также делает код более гибким, приспособленным к изменениям и упрощает добавление новых возможностей. Кроме того, такой код легче понимать и поддерживать.

Примеры использования ООП можно найти в создании бизнес-программ, веб-приложений, игр, мобильных приложений и систем искусственного интеллекта. Это важный и распространенный подход, позволяющий создавать надежные и сложные программы, и каждому современному программисту нужно знать основы ООП.

Пример тестового задания.
Что такое класс в ООП?
a) Это экземпляр объекта.
b) Это функция или процедура.
c) Это переменная, хранящая данные.
d) Это шаблон или чертеж для создания объектов.

Разбор.
a) Это экземпляр объекта. Неверно. Экземпляр объекта – это объект, а не класс. Класс существует до создания объекта.
b) Это функция или процедура. Неверно. Класс может содержать функции (методы), но сам класс не является функцией или процедурой.
c) Это переменная, хранящая данные. Неверно. Класс определяет структуру данных, которые будут храниться в объектах этого класса, но сам класс не является переменной.
d) Это шаблон или чертеж для создания объектов. Верно. Класс определяет структуру (атрибуты) и поведение (методы), которые будут иметь объекты этого класса. Он служит планом для создания новых объектов.

Вывод. Исходя из вопроса и предложенных ответов, правильным ответом является - d) Это шаблон или чертеж для создания объектов.

Раздел 7. Работа в офисных программах

Вопрос 1. Как создать новый документ в Microsoft Word?
— Нажать Ctrl+N
— Нажать Ctrl+P
— Нажать Ctrl+O

Ответ: Нажать Ctrl+N

Вопрос 2. Как вставить таблицу в Microsoft Excel?
— В меню «Вставка» выбрать «Таблица»
— В меню «Формат» выбрать «Таблица»
— В меню «Редактирование» выбрать «Таблица»

Ответ: В меню «Формат» выбрать «Таблица»

Вопрос 3. Как добавить новый слайд в Microsoft PowerPoint?
— Нажать Ctrl+M
— Нажать Ctrl+N
— Нажать Ctrl+O

Ответ: Нажать Ctrl+M

Раздел 8. Теоретические знания по информатике

В XXI веке скорость распространения и объем хранимой информации стали практически неограниченными. Более того, существенная доля человеческого труда сегодня затрачивается не на производство материальных благ, а на обработку информации!

Именно потому, что производство и потребление информации составляет существенную долю в жизни нашего общества, оно и называется «информационным». Сегодня человек может совершить кругосветное путешествие, посетить другие планеты, увидеть далекие звезды и получить ответы ны любые интересующие его вопросы не выходя из дома.

Мы можем гулять по вымышленным мирам населенным эльфами или киборгами-мутантами. Там мы становимся кем-то другим, не тем, кто мы есть в повседневной жизни. Но у заманчивых благ информационного общества есть и обратная сторона. Устрашающим последствием информатизации является невиданная доселе прозрачность личности.

Данные о каждом человеке собираются кредтиными организациями, сотовыми операторами связи, налоговыми органами и, конечно, спецслужбами. Таким образом, для постороннего вмешательства в личную жизнь человека открываются самые широкие перспективы. Такие, какие и не снились тоталитарным режимам прошлого! Обилие и доступность информации бумерангом ударило по современному человеку. На нас ежедневно обрушивается «информационный шквал»: телевидение, реклама, городской шум, Интернет и пр. Более того, пути, которыми мы сегодня

получаем информацию отличаются от тех, для которых наши органы чувств сформировала эволюция.

Информатику удобно подразделить на:

Теоретическую (раздел прикладной математики)

Теоретическая информатика разрабатывает математический аппарат исследования процессов хранения, обработки и передачи информации.

Прикладную (разделы прикладной математики, физики, психологии, биологии и пр.)

Прикладная Информатика в отличие от Теоретической, не ставит своей целью развитие математического аппарата Информатики.

Прикладная информатика использует математический аппарат Информатики, а также достижения и методы других наук для изучения процессов хранения, обработки и передачи информации.

Практическую (Создание прикладных программ)

Практическая информатика относится к инженерным специальностям. Специалисты в области практической информатики занимаются разработкой программного обеспечения, используя достижения Теоретической и Прикладной информатики.

Техническую (раздел техники и технологии, инженерные задачи) Техническая информатика – это тоже инженерная сфера человеческой деятельности. Информатике относят вопросы конкретной реализации материальной базы (hardware) процессов хранения, обработки и передачи информации.

Примеры заданий:

Вопрос 1. Сколько в 1 Килобайт бит?

Ответ 2^{10}

Вопрос 2. Напишите имя учёного, который ввел определение энтропии – меры неопределнности

Ответ: Клод Шеннон

Вопрос 3. Как называется десятичная единица измерения объема информации.

Ответ: дит

Литература для подготовки

1. Информатика: 10-й класс: учебник. Босова Л.Л., Босова А.Ю., АО «Издательство «Просвещение».
2. Информатика: 10-й класс: учебник. Гейн А.Г. АО «Издательство «Просвещение»

Информационные ресурсы:

1 Единая коллекция цифровых образовательных ресурсов [Электронный ресурс]. – Режим доступа http://school-collection.edu.ru

2 Сайт журнала информатика. [Электронный ресурс]. – Режим доступа Журнал «Информатика и образование» — Издательство "Образование и Информатика"