Билет 39.

Рекурсия Функция, которая прямо или косвенно вызывает сама себя, называется рекурсивной. Если в теле функции явно используется вызов той же самой функции, то имеет место прямая рекурсия (self-calling). Если две или более функций взаимно вызывают друг друга, то имеет место косвенная рекурсия.

Билет 40.

Символьный тип Данные типа char в памяти компьютера всегда занимают 1 байт. Символьный тип может быть со знаком или без него. В величинах со знаком signed char можно хранить значение от - 128 до 127. Соответственно значения переменных типа unsigned char могут находиться в диапазоне от 0 до 255. При работе с символьными данными нужно помнить, что если в выражении встречается одиночный символ, то он должен быть заключен в одинарные кавычки (‘a’). Символ – элементарная единица, некоторый набор которых несет определенный смысл. В языке программирования С++ предусмотрено использование символьных констант. Символьная константа – это целочисленное значение (типа int) представленное в виде символа, заключённого в одинарные кавычки, например 'a'. В таблице ASCII представлены символы и их целочисленные значения. 1 // объявления символьной переменной 2 char symbol = 'a'; // где symbol – имя переменной типа char // char – тип данных для хранения символов Строки в С++ представляются к ак массивы элементов типа char, заканчивающиеся нультерминатором \0 называются С строками или строками в стиле С. \0 — символ нуль-терминатора. Символьные строки состоят из набора символьных констант заключённых в двойные кавычки. При объявлении строкового массива необходимо учитывать наличие в конце строки нультерминатора, и отводить дополнительный байт под него. // пример объявления строки char string[10]; // где string – имя строковой переменной // 10 – размер массива, то есть в данной строке может поместиться 9 символов , последнее м Строка при объявлении может быть инициализирована начальным значением, например, так: 1 char string[10] = "abcdefghf"; Если подсчитать кол-во символов в двойных кавычках после символа равно их окажется 9, а размер строки 10 символов, последнее место отводится под нуль–терминатор, причём компилятор сам добавит его в конец строки. // посимвольная инициализация строки: char string[10] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'f', '\0'}; // десятый символ это нуль-терминатор. При объявлении строки не обязательно указывать её размер, но при этом обязательно нужно её инициализировать начальным значением. Тогда размер строки определится автоматически и в конец строки добавится нуль-терминатор. 1 //инициализация строки без указания размера 2 char string[] = "abcdefghf"; 3 //всё то же самое только размер не указываем. Строка может содержать символы, цифры и специальные знаки. В С++ строки заключаются в двойные кавычки. Имя строки является константным указателем на первый символ.

Билет 10

Форматированный ввод данных. Форматированный ввод данных осуществляется с помощью оператора «scanf». Шаблон ввода: %[\*][Длинна][модификатор] символ преобразования.

Билет 28

Побитовые операции. & битовое И (AND) | битовое ИЛИ (OR) ^ битовое ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR) ~ битовое ОТРИЦАНИЕ (NOT) - унарный оператор Операция поразрядного логического И (&) сравнивает каждый бит первого операнда с соответствующим битом второго операнда. Если оба сравниваемых бита единицы, то соответствующий бит результата устанавливается в 1, в противном случае в 0. Операция поразрядного логического ИЛИ (|) сравнивает каждый бит первого операнда с соответствующим битом второго операнда. Если любой (или оба) из сравниваемых битов равен 1, то соответствующий бит результата устанавливается в 1, в противном случае результирующий бит равен 0. Операция поразрядного исключающего ИЛИ (^) сравнивает каждый бит первого операнда с соответствующими битами второго операнда. Если один из сравниваемых битов равен 0, а второй бит равен 1, то соответствующий бит результата устанавливается в 1, в противном случае, т.е. когда оба бита равны 1 или 0, бит результата устанавливается в 0.

Билет 31

Операции отрицания и побитового дополнения. Побитовое отрицание (или побитовое НЕ, или дополнение) — это , действие которой эквивалентно применению логического отрицания к каждому биту двоичного представления операнда. Другими словами, на той позиции, где в двоичном представлении операнда был 0, в результате будет 1, и, наоборот, где была 1, там будет 0. Например: НЕ 01 10 Операции отрицания Перечислим операции отрицания: -, !, ~ Арифметическое отрицание (унарный минус - ) изменяет знак операнда целого или вещественного типа на противоположный. Логическое отрицание (!) дает в результате значение О, если операнд есть истина (не нуль), и значение 1, если операнд равен нулю. Операнд должен быть целого или вещественного типа, а может иметь также тип указатель. Поразрядное отрицание (~), называют побитовым, инвертирует каждый разряд в двоичном представлении целочисленного операнда.

Билет 21

Оператор цикла с постусловием Циклический алгоритм с постусловием: цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз. Структура оператора цикла с постусловием: do{}while()

Билет 44.

Низходящий метод Нисходящее проектирование – подход к проектированию программ, при котором первоначально создается главный модуль, для которого затем проводится декомпозиция (разбиение на модули, решающие подзадачи главной задачи). Метод нисходящего проектирования предполагает последовательное разложение общей функции обработки данных на простые функциональные элементы ("сверхувниз"). В результате строится иерархическая схема, отражающая состав и взаимоподчиненность отдельных функций. Нисходящее проектирование. по алгоритму программы создается текст главного модуля, содержащего в виде комментариев основны6е этапы решения задачи. Желательно, чтобы каждая строка комментария представляла собой вызов подпрограммы, решающей подзадачу главной задачи основные этапы заменяются строчкой кода, представляющего собой вызов функции, соответствующей этапу решения задачи. Выбирается имя функции, тип возвращаемого значения и определяется список параметров. По каждой вновь созданной1 функции записывается определение. Тело функции содержит комментарии. Процедура повторяется до тех пор, пока комментарии не будут заменены кодом на языке С

Билет 13

Структура простой программы. Операция инкремента и декремента. Структура простой программы: Программа, записанная на языке Си состоит из: 1) Совокупности функций, одна из которых функция main(). 2) Описание функции состоит из заголовка и тела функции. 2.1) Заголовок содержит операторы препроцессора, например, директиву #include . 2.2) Тело функции заключается в фигурные скобки и состоит из ряда операторов, каждый из которых заканчивается ";". Операция инкремента и декремента: Cсинтаксис использования операций инкремента и декремента таков, что перед или после имени переменной ставится операция инкремента или декремента. Когда операция инкремента или декремента ставится перед именем переменной, то такая операция называется префиксным инкрементом (сокращённо — преинкрементом) или префиксным декрементом (сокращённо — предекрементом). А если операция инкремента или декремента ставится после имени переменной, то такая операция называется операцией постфиксного инкремента (сокращённо — постинкремент) или постфиксного декремента (сокращённо — постдекремент). При использовании операции преинкремента значение переменной, сначала, увеличивается на 1, а затем используется в выражении. При использовании операции постинкремента значение переменной сначала используется в выражении, а потом увеличивается на 1. При использовании операции предекремента, значение переменной, сначала, уменьшается на 1, а затем используется в выражении. При использовании операции постдекремента, значение переменной, сначала, используется в выражении, а потом уменьшается на 1

Билет 5

ИТЕРАЦИОННЫЙ ЦИКЛ Оператор цикла, для которого число повторений тела цикла заранее неизвестно. В итерационных циклах накаждом шаге вычислений происходит последовательное приближение и пр оверка условия достиженияискомого результата. Выход из ите рационного цикла осуществляется в случае выполнения задан ного условия. Пример итерационного цикла цикл for в C/C++

Билет 22

Оператор цикла с предусловием. Составной и пустой операторы. Циклический алгоритм с предусловием: цикл, который выполняется пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно). Составной оператор: Синтаксис: { [объявление] . . . оператор [оператор] . . . } Выполнение Составной оператор обычно появляется как тело другого оператора, например, оператора if. При выполнении составного оператора его операторы выполняются в том порядке, в котором они появляются, если не встретится оператор, который явно передает управление в другое место программы. Пустой оператор: состоит только из точки с запятой. При выполнении этого оператора ничего не происходит. Он обычно используется в следующих случаях: - в операторах do, for, while, if в строках, когда место оператора не требуется, но по синтаксису требуется хотя бы один оператор; - при необходимости пометить фигурную скобку.

Билет 4

Циклические алгоритмы, управляемые условием. Циклический алгоритм с предусловием: цикл, который выполняется пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно). Циклический алгоритм с постусловием: цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

Билет 29

Логические операции. В С++ существует три логические операции: Логическая операция И &&, нам уже известная; Логическая операция ИЛИ ||; Логическая операция НЕ ! или логическое отрицание. Логические операции образуют сложное (составное) условие из нескольких простых (два или более) условий. Эти операции упрощают структуру программного кода в несколько раз. Да, можно обойтись и без них, но тогда количество ифов увеличивается в несколько раз, в зависимости от условия. В следующей таблице кратко охарактеризованы все логические операции в языке программирования С++, для построения логических условий.

Билет 19

Оператор переключатель switch. Комментарии Оператор переключатель switch: Синтаксис: switch() { [] [case :] [] [case :] [] [default:] [] } Действие: Оператор-переключатель switch предназначен для выбора одного из нескольких альтернативных путей выполнения программы. Выполнение оператора-переключателя начинается с вычисления значения выражения переключения (выражения, следующего за ключевым словом switch в круглых скобках). После этого управление передается одному из тела переключателя. В теле переключателя содержатся конструкции case :, которые синтаксически представляют собой метки операторов.. Оператор, получающий управление, — это тот оператор, значение константы варианта которого совпадает со значением выражения переключения. Значение каждой константы варианта должно быть уникальным внутри тела оператора-переключателя. Выполнение тела оператора-переключателя switch начинается с выбранного таким образом оператора и продолжается до конца тела или до тех пор, пока какой-либо оператор не передаст управление за пределы тела. Оператор, следующий за ключевым словом default, выполняется, если ни одна из констант варианта не равна значению выражения переключения. Если же слово default опущено, то ни один оператор в теле переключателя не выполняется, и управление передается на оператор, следующий за переключателем в программе. Комментарии: пояснения к исходному тексту программы, находящиеся непосредственно внутри комментируемого кода. Комментарии не оказывают никакого влияния на результат компиляции программы или её интерпретацию. Помимо исходных текстов программ, комментарии также применяются в языках разметки и языках описания.

Билет 36.

Определение функции Определение функции должно располагаться в глобальной области видимости, до начала функции main. Определение функции состоит из заголовка и тела. Заголовок фукнции включает в себя: Тип возвращаемого значения Почти все функции должны возвращать значения. Тип этого значения указывается в заголовке перед именем функции. Вот несколько примеров заголовков функций: ). Возвращаемые значения используются для передачи данных из функции в вызывающее окружение. Вызывающее окружение - это то место, откуда вызывается данная функция, подробнее ниже. Идентификатор или имя функции Идентификатор (имя) функции задаётся точно также, как и любой другой идентификатор. Список аргументов или параметров Список аргументов функции записывается в круглых скобках после имени функции. В данном примере список аргументов пуст. Список аргументов записывается через запятую. Каждый элемент списка состоит из типа и идентификатора. Рассмотрим пример заголовка функции со списком из двух аргументов: int simple (int a, float b) В скобках мы записали два аргумента: a и b. У аргумента a тип int, а у аргумента b тип float. Аргументы используются, когда в функцию нужно передать какие-либо данные из вызывающего окружения. Вызов функции состоит из идентификатора функции и списка аргументов в круглых скобках. Вот несколько вызовов функции sum:

Билет 46.

Порядок выполнения программы на пк Для того, чтобы выполнить программу на С++, надо пройти шесть этапов: Первый этап - редактирование; Второй этап - предварительная (препроцессорная) обработка; Третий этап - компиляция; Четвертый этап - компоновка; Пятый этап - загрузка; Шестой этап - выполнение. Редактирование. Это первый этап разработки программы в среде программирования и представляет он собой редактирование файла (исходного файла, который в последствии будет содержать код программы). Предварительная (препроцессорная) обработка. На этом этапе программист дает команду компилироватьпрограмму. Но прежде чем компилятор приступит к компиляции вашей программы, производится предварительная обработка программы. Компиляция. На этом этапе компилятором проверяется текст программы на наличие синтаксических ошибок и затем, если все хорошо, текст программы с подстановками, сделанными на предыдущем этапе, преобразуется в машинный код (код на языке, уже непосредственно понятный компьютеру). Компоновщик связывает объектный код с кодами отсутствующих функций, чтобы создать исполняемый загрузочный модуль (без пропущенных частей). Загрузка. Следующий этап называется загрузка. Перед выполнением программа должна быть размещена в оперативной памяти компьютера. Это делается с помощью загрузчика Выполнение. И наконец, рассмотрим самый последний этап - выполнение. С этого момента компьютер под управлением своего ЦПУ (центральное процессорное устройство) начинает последовательно выполнять в каждый момент времени по одной команде программы. Трансля́тор - Машинная программа, которая транслирует с одного языка на другой и, в частности, с одного языка программирования на другой. Цель трансляции — преобразовать текст с одного языка на другой, который понятен адресату текста. В случае программтрансляторов, адресатом является техническое устройство (процессор) или программа-интерпретатор. Интерпрета́тор (языка программирования) - Программа (иногда аппаратное средство), анализирующая команды или операторы программы и тут же выполняющая их. Простой интерпретатор анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора.

Билет 17

Операции отношения и сдвига. Операция sizeof. Операция сдвига: Операции сдвига сдвигают свой первый операнд влево (<>) на число разрядов машинного слова. Оба операнда должны быть целыми значениями. Операция отношения: "<" - меньше. "<=" - меньше или равно. ">" - больше. ">=" - больше или равно. "==" - равно. "!=" - не равно. Операция sizeof: Эта операция выполняется на стадии компиляции. Результатом этой операции является число байтов, необходимое для размещения объекта в памяти. Существует два варианта синтаксиса этой операции. В первом из них единственный операнд операции определяет некоторый тип языка, и он должен быть заключен в скобки: sizeof ( float ) sizeof ( int ) Во втором операнд задает некоторое выражение и здесь использование скобок необязательно: sizeof a; sizeof \*ip; sizeof array[ i ];

Билет 14

Оператор «выражение». Приоритет операций. Стандартные функции. Оператор выражение: включает в себя выражение, за которым следует точка с запятой. Приоритет операций: Приоритет операций - очерёдность выполнения операций в выражении, при условии, что в выражении нет явного указания порядка следования выполнения операций (с помощью круглых скобок) Если операции имеют одинаковый приоритет, то очерёдность выполнения таких операций определяется согласно свойству ассоциативности. Ассоциативность - направление выполнения операций в случае, если операции имеют одинаковый приоритет. Стандартные функции: 1 условные(if swith) 2 цикла(for while do-while) 3 операторы переходов (goto break continue return) 4др операторы(присваивания пустой составной) или математические операции присваивание составное присваивание преобразование типов операции отношения операции инкремента (++) и декремента (-- операция sizeof операция <> оператор ?: указатель; определение адреса переменной

Билет 34

Подпрограммы. Локальные и глобальные параметры Глобальные переменные Переменная, объявленная на внешнем уровне, имеет глобальное время жизни. При отсутствии инициализатора такая переменная инициализируется нулевым значением. Область видимости переменной, определенной на внешнем уровне, распространяется от точки, где она определена, до конца исходного файла. Переменная недоступна выше своего определения в том же самом исходном файле. На другие исходные файлы программы область видимости переменной может быть распространена только в том случае, если ее определение не содержит спецификации класса памяти static. Если в объявлении переменной задана спецификация класса памяти static, то в других исходных файлах могут быть определены другие переменные с тем же именем и любым классом памяти. Эти переменные никак не буду связаны между собой. Спецификация класса памяти extern используется для объявления переменной, определенной где-то в другом месте программы. Такие объявления используются в случае, когда нужно распространить на данный исходный файл область видимости переменной, определенной в другом исходном файле навнешнем уровне. Область видимости переменной распространяется от места объявления до конца исходного файла. В объявлениях, которые используют спецификацию класса памяти extern, инициализация не допускается, так как они ссылаются на переменные, значения которых определены в другом месте. 1.1.2. Локальные переменные Переменная, объявленная на внутреннем уровне, доступна только в том, блоке в котором она объявлена, независимо от класса памяти. По умолчанию она имеет класс памяти auto. Переменные этого класса размещаются в стеке. Переменные класса памяти auto автоматически не инициализируются, поэтому в случае отсутствия инициализации в объявлении значение переменной класса памяти auto считается неопределенным. Спецификация класса памяти register требует, чтобы переменной была выделена память в регистре, если это возможно. Т.к. работа с регистрами происходит быстрее, спецификация класса памяти register обычно используется для переменных, к которым предполагается обращаться очень часто. Для каждого рекурсивного входа в блок порождается новый набор переменных класса памяти autoи register. При этом каждый раз производится инициализация переменных, в объявлении которых заданы инициализаторы. Если переменная, объявленная на внутреннем уровне, имеет спецификацию памяти static, то область видимости остается прежней, а время жизни становится глобальным. В отличие от переменных класса памяти auto, переменные, объявленные со спецификацией класса памяти static, сохраняют свое значение при выходе из блока. Переменные класса памяти static могут быть инициализированы константным выражением. Если явной инициализации нет, то переменная класса памяти staticавтоматически инициализируется нулевым значением. Инициализация выполняется один раз во время компиляции и не повторяется при каждом входе в бока. Все рекурсивные вызовы данного бока будут разделять единственный экземпляр переменной класса памяти static. Переменная, объявленная со спецификацией класса памяти extern, является ссылкой на переменную с тем же самым именем, определенную на внешнем уровне в любом исходном файле программы. Цель внутреннего объявления extern состоит в том, чтобы сделать определение переменной доступным именно внутри данного блока

Билет 1

Алгоритм. Свойства алгоритма. Способ записи алгоритма. Алгоритм - это последовательность арифметических и логических действий над данными, приводящая к получению решения поставленной задачи. Свойства: А) Дискретность - алгоритм состоит из отдельных пунктов или шагов Б) Определённость - каждый шаг алгоритма должен иметь точный смысл. В) Связанность - на каждом следующем шаге используются результаты предыдущего. Г) Конечность – алгоритм должен завершаться после конечного числа шагов Д) Результативность – алгоритм должен приводить к получению конечных результатов Е) Массовость – пригодность для решения широкого класса задач. Ж) Эффективность – применение а. должно давать какой то положительный временной результат (временной) Способ записи: А) Словесно-формульный Б) Структурная схема и алгоритм (ССА) В) Спец. языки (алгоритмические и псевдокоды) (псевдокод - искусственный неформальный язык, обычно состоит из элементов обычного языка с элементами программирования) Г) Графический способ

Билет 9

Форматированный вывод данных. Форматированный вывод данных осуществляется с помощью оператора «printf». Шаблон вывода: %[флажок][длина][.точность][модификатор]символ преобразования