Lesson\_3

|  |  |
| --- | --- |
| Здравствуйте коллеги. Рад всех приветствовать на третьем уроке по основам языка С. На этом занятии мы рассмотрим такие ключевые конструкции языка как "условные операторы", а также что понимается под "блоками кода" и "областями видимости". |  |
| Пожалуй, самый часто используемый в любом языке программирования, в том числе и в языке С оператор - это оператор if() и его вариации. Оператор if позволяет программе принять решение о выполнении или невыполнении того или иного действия в зависимости от текущего состояния. Например, зададим пользователю вопрос, хочет ли он, чтобы его поприветствовали, *для этого опишем переменную char answer, которая будет хранить ответ пользователя и спросим у пользователя в консоли хочет ли он чтобы мы его поприветствовали, для этого используем функцию printf, которой в качестве параметра передадим строку. Далее при помощи функции scanf считаем ответ пользователя в переменную answer и в зависимости от пользовательского ввода программа либо поприветствует пользователя, либо нет, это решение будет принято с помощью оператора if.* В случае, если условие в скобках выполнится, выполнится и последующий код в фигурных скобках. Если условие не выполнится, то все операторы внутри этих фигурных скобок будут проигнорированы. | if()  char answer;  printf("do you want me to salute you (y/n)?");  scanf ("%s", &answer);  if(answer == "y") {  printf("Hello, user");  } |
| Зачастую складываются ситуации, когда нужно выполнить разные наборы действий в зависимости от результата проверки условия. Для таких случаев используется дополнение к оператору if - оператор else в котором описывается последовательность действий выполняемая в случае если условие в скобках дало ложный результат. Как вы видите, в зависимости от того что ввел пользователь мы реализуем ту или иную ветку оператора if-else. Конструкция if-else является единым оператором выбора. | else{  printf("I didn't want to salute you anyway");  } |
| Множественный выбор при помощи оператора if можно осуществить используя конструкцию if-else if-else. Данное усложнение также будет являться единым оператором выбора. *Добавим в нашу конструкцию еще одно условие if и опишем пользовательский ввод для ответа “да” и ответа “нет”. В этом примере оператором else будет непонимание программы того что ввел пользователь. Выведем в консоль надпись «Я не могу понять ваш ввод».*  Операторов else if в одной управляющей конструкции может быть сколько угодно, в отличие от оператора if и оператора else которых не может быть больше одного. | if(answer == "y") {  printf("Hello, user");  } else if (answer == "n"){  printf("I didn't want to salute you anyway");  } else {  printf("I can't understand your input");  } |
| Для короткой или внутристрочной записи условного оператора, а также для присваивания переменных по условию можно использовать тернарный оператор. Например, создадим две целочисленные переменные а, b и зададим им какие-нибудь начальные значения, допустим а = 10 и b = 15 и запись вида a = (a > b) ? b : a будет обозначать, что в случае если a > b в переменную а запишется значение b, и наоборот если b > a, то в переменную а запишется значение а. *Давайте распишем это в привычном виде if-else. Поскольку в такой конструкции в ветке else будет происходить присваивание переменной а переменной а, что попросту не имеет смысла, мы можем удалить эти строки. Таким образом, получим короткую запись без использования тернарного оператора.* | int a;  int b;  a = (a > b) ? b : a  if (a > b){  a = b;  } ~~else {~~  ~~a = a;~~  ~~}~~ |
| Также тернарный оператор можно использовать для удобного форматированного вывода, *например опишем функцию printf , которая будет печатать нам строку и в зависимости от условия это будет "true" либо "false". Давайте проверим как это работает, в результате видим true, потому что единица действительно больше ноля.* | printf("\n%s", (1>0) ? "true" : "false"); |
| *Закомментируем кусок кода с приветствием и поговорим об операциях сравнения.* Операции сравнения бывают трёх типов - арифметические, логические и тернарный оператор. Арифметическое сравнение - это привычные нам с детства операторы больше, меньше, больше или равно, меньше или равно и проверка на равенство, которая записывается в виде двух знаков равенства. Возвращают истину, когда выполняются соответствующие названиям условия.  Логических операторов четыре - это И, ИЛИ, НЕ и ИСКЛЮЧАЮЩЕЕ ИЛИ. В отличие от арифметических двоичных операторов - логические возвращают истину и ложь т.е. 1 либо 0.  Оператор И возвращает истину когда оба операнда истинны  Оператор ИЛИ возвращает истину когда хотя бы один из операндов истинный  Оператор НЕ возвращает истину когда операнд ложный  Оператор ИСКЛЮЧАЮЩЕГО ИЛИ возвращает истину когда операнды различаются  *Обращаю ваше внимание, что если в операторе if будет выполнятся только одно действие, фигурных скобок может и не быть. Используя логические операторы в программе мы можем написать логику практически любой сложности* | // > < >= <= ==  // && || ! ^  if (1 && 0) printf("false");  if (1 || 0) printf("true");  if (!(1)) printf("false");  if (1 ^ 1) printf("false"); |
| Внутри операторных скобок конструкции if могут находиться другие условные операторы. Условия, расположенные таким образом называются вложенными. Никаких ограничений на использование вложенных условий в языке С нет. *Закомментируем этот кусок кода, чтобы он не мешал.* | /\* if (1) {  if (1){  //...  }  }  \*/ |
| В языке Си также нет ограничений на использование сложных условий. Сложные условия это такие условия, где в круглых скобках выполняется более одного сравнения. Сравнения производятся в порядке заранее оговоренного приоритета*.* |  |
| Например, дана некоторая переменная int x = 7 нужно выяснить, не выходит ли эта переменная за рамки заданных значений. *Запишем условие: если x >= 0 и x <= 10 выдать сообщение о том, что х подходит. Добавим новую строку, чтобы отделить этот пример и рассмотрим данную запись подробнее.*  В данной записи мы видим, что сначала х сравнивается с нулём, затем с десятью, и в конце результаты будут сравнены между собой.  Самый неприоритетный оператор тернарный. Если Вы сомневаетесь в приоритете сравнений или Вам необходимо описать какое-то очень сложное условие, всегда можно воспользоваться скобками, задав приоритет операций явно. В таком случае в первую очередь будут выполнены операции в скобках.  Для того, чтобы пользоваться арифметическими бинарными операторами в качестве логических, обязательно нужно удостовериться, что вы сравниваете нули и единицы, а не какие-то другие числа. | printf("\n");  int x = 7;  if ((x >= 0) && (x <= 10)) {  printf("X Fits!");  } |
| Говоря об операторах языка С и управляющих конструкциях нельзя не сказать о "блоках кода" и "областях видимости". Как видно, условные операторы содержат записи в фигурных скобках. В такие же скобки заключён код функции main. Эти скобки называются "операторными", а то что в них содержится, называется "блоком кода". Все переменные, которые инициализируются внутри блока кода, существуют и видны только внутри кодового блока. Поэтому пространство между операторными скобками также называют "областью видимости*". Итак вы видите что мы не можем напечатать значение переменной var, поскольку она была создана внутри блока кода оператора if и перестала существовать как только мы вышли за его пределы. Удалим инициализацию и вывод в консоль переменной var, запустим нашу программу и удостоверимся что все работает.* | printf("\n");  int x = 7;  if ((x >= 0) && (x <= 10)) {  ~~int var = 0;~~  printf("X Fits!");  }  ~~printf("\n%d", var);~~ |
| Поговорим немного о Булевой алгебре и двоичных вычисления?  Булева алгебра это один из базовых, но вместе с тем один из самых мощных инструментов в программировании. Двоичные вычисления выполняются быстрее десятичных, поскольку являются естественными для цифровой техники. В бинарной алгебре используются операторы И, ИЛИ, НЕ, ИСКЛЮЧАЮЩЕЕ ИЛИ и операции СДВИГА влево и вправо. Работают эти операторы точно также, как и логические, но относительно разрядов двоичного представления чисел, где истина – это единица, а ложь - это ноль. Условия истинности точно такие же как и для логических операторов, повторим:  Оператор И возвращает единицу когда оба операнда единицы  Оператор ИЛИ возвращает единицу когда хотя бы один из операндов единица  Оператор НЕ возвращает единицу когда операнд равен нулю  Оператор ИСКЛЮЧАЮЩЕГО ИЛИ возвращает единицу когда операнды различаются | printf("\n");  // & | ! ^ << >>  /\*  \* 1 & 0 = 0  \* 1 | 0 = 1  \* !1 = 0  \* 0 ^ 1 = 1  \*/ |
| На основе этих знаний мы можем для примера написать программу, меняющую местами значения переменных без использования третьей, вспомогательной. *Объявим две переменных a и b, присвоим им значения и выведем их в консоль. Закомментируем предыдущий кусок кода чтобы не сбивать инициализацию наших переменных. Далее подготовим вывод измененных значений a и b в консоль* printf("a = %d, b = %d", a, b);  *и напишем некую магическую конструкцию*. *В переменную а нужно будет записать a ^ b, в переменную b нужно будет записать b ^ a и в переменную а нужно будет записать a = a ^ b.*  Рассмотрим происходящее пошагово  Число 11 типа char в двоичном представлении это 00001011, число 15 - это 00001111. После выполнения первого оператора в переменную а будет положено промежуточное число 00000100 – это цифра 4. Оператор ксор выполняется следующим образом – результат будет равен 1 если операнды различаются и 0 если они совпадают.  После выполнения второго в переменную b будет положено число 00001011. Здесь b = 15 (00001111), a = 4 (00000100), производим операцию исключающего ИЛИ и получаем 00001011 – т.е. цифру 11.  И после выполнения третьего в переменную а будет положено значение 00001111 – это цифра 15.  Выведем значения переменных в консоль и убедимся в правильности нашего решения. | char a = 11;  char b = 15;  printf("a = %d, b = %d", a, b);  a = a ^ b;  b = b ^ a;  a = a ^ b;  printf("a = %d, b = %d", a, b);  a = a ^ b; //00000100  // a = 11 (00001011)  // b = 15 (00001111)  // a ^ b = 00000100  b = b ^ a; // b = 00001011  // b = 15 (00001111)  // a = 4 (00000100)  // a ^ b = 00001011  a = a ^ b; // a = 00001111  // a = 4 (00000100)  // b = 11 (00001011)  // a ^ b = 00001111  printf("a = %d, b = %d\n", a, b); |
| Операции сдвига бывают логические, арифметические и циклические. В языке С реализован логический сдвиг, то есть недостающие разряды заполняются нулями.  Сдвиг влево на n значений это умножение на 2 в степени n.  Сдвиг вправо на n значений это целочисленное деление на 2 в степени n.  *Итак, нам нужно переменную а сдвинуть влево на 3, на самом деле это означает что мы переменную а умножим на 2 в степени 3. А переменную b мы сдвинем вправо на 2 – это означает, что мы переменную b разделим на 2 в степени 2. Это тоже самое что записать переменная а\*8 и b/4. Просто выполнится это гораздо быстрее.*  Бинарная алгебра это большая и сложная тема. Применять бинарную алгебру можно и в высокоуровневых проектах, поддержка бинарных операций есть в подавляющем числе языков программирования. | a = a << 3; // 15 \* 8  b = b >> 2; // 11 / 4  printf("a = %d, b = %d", a, b); |
| На следующем уроке мы начнём говорить о циклах. До новых встреч коллеги! |  |