Тема : знакомство с C (Си)
SWITCH, BREAK, FOR, WHILE, DO WHILE, CONTINUE, упражнения

Содержание: SWITCH BREAK FOR WHILE DO WHILE CONTINUE упражнения

• Оператор SWITCH

Оператор **switch** предназначен для организации выбора из множества различных вариантов. Формат оператора следующий:

```
     switch ( выражение )
     {    [объявление]
                :
 [ case  константное-выражение1]: [ список-операторов1]
 [ case  константное-выражение2]: [ список-операторов2]
                :
                :
 [ default: [ список операторов ]]
     }
```

Выражение, следующее за ключевым словом switch в круглых скобках, может быть любым выражением, допустимыми в языке **C**, значение которого должно быть целым.

Значение этого выражения является ключевым для выбора из нескольких вариантов. Тело оператора **switch** состоит из нескольких операторов, помеченных ключевым словом **case** с последующим константным-выражением. Следует отметить, что использование целого константного выражения является существенным недостатком, присущим рассмотренному оператору**.**

Так как константное выражение вычисляется во время трансляции, оно не может содержать переменные или вызовы функций. Обычно в качестве константного выражения используются целые или символьные константы.

Все константные выражения в операторе **switch** должны быть уникальны. Кроме операторов, помеченных ключевым словом **case**, может быть, но обязательно один, фрагмент помеченный ключевым словом **default**.

Список операторов может быть пустым, либо содержать один или более операторов. Причем в операторе **switch** не требуется заключать последовательность операторов в фигурные скобки.

Схема выполнения оператора **switch** следующая:

- вычисляется **выражение** в круглых скобках;
- вычисленные значения последовательно сравниваются с константными выражениями, следующими за ключевыми словами **case**;

- если одно из константных выражений совпадает со значением выражения, то управление передается на оператор, помеченный соответствующим ключевым словом **case**;

- если ни одно из константных выражений не равно выражению, то управление передается на оператор, помеченный ключевым словом **default,** а в случае его отсутствия управление передается на следующий после **switch** оператор.

Для того, чтобы выполнить одни и те же действия для различных значений выражения, можно пометить один и тот же оператор несколькими ключевыми словами **case**.

Пример:

```
int i=2;
switch (i)
  {
   case 1: i += 2;
   case 2: i *= 3;
   case 0: i /= 2;
   case 4: i -= 5;
   default:      ;
  }
```

Выполнение оператора **switch** начинается с оператора, помеченного **case 2**. Таким образом, переменная **i** получает значение, равное 6, далее выполняется оператор, помеченный ключевым словом **case 0**, а затем **case 4**, переменная **i** примет значение 3, а затем значение -2. Оператор, помеченный ключевым словом **default**, не изменяет значения переменной. (*Здесь* запись, например, `i += 2; эквивалентна записи i = i+2;`)

Пример:

```
char ZNAC;
   int x,y,z;
   switch (ZNAC)
   {
        case '+':  x = y + z;    break;
        case '-':  x = y - z;    break;
        case '*':  x = y * z;    break;
        case '/':  x = u / z;    break;
        default : ;
    }
```

Использование оператора **break** позволяет в необходимый момент прервать последовательность выполняемых операторов в теле оператора **switch,** путем передачи управления оператору, следующему за **switch.**

Отметим, что в теле оператора **switch** можно использовать вложенные операторы **switch**, при этом в ключевых словах **case** можно использовать одинаковые константные выражения.

Пример:

```
switch (a)
    {
      case 1: b=c; break;
      case 2:
          switch (d)
          {   case 0:  f=s;  break;
              case 1:  f=9;  break;
              case 2:  f-=9; break;
          }
       case 3: b-=c; break;
       …
    }
```

- ## Оператор BREAK

Оператор **break** обеспечивает прекращение выполнения самого внутреннего из объединяющих его операторов **switch**, **do**, **for**, **while**. После выполнения оператора **break** управление передается оператору, следующему за прерванным.

- ## Оператор FOR

Оператор **for** – это наиболее общий способ организации цикла. Он имеет следующий формат:

for (выражение 1 ; выражение 2 ; выражение 3) тело

Выражение 1 обычно используется для установления начального значения переменных, управляющих циклом. Выражение 2 – это выражение, определяющее условие, при котором тело цикла будет выполняться. Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

Схема выполнения оператора **for:**

1. Вычисляется **выражение 1.**

2. Вычисляется **выражение 2.**

3. Если значения **выражения 2** отлично от нуля (истина), выполняется **тело** цикла, вычисляется **выражение 3** и осуществляется переход к пункту 2, если **выражение 2** равно нулю (ложь), то управление передается на оператор, следующий за оператором **for.**

Существенно то, что проверка условия всегда выполняется в начале цикла. Это значит, что тело цикла может ни разу не выполниться, если условие выполнения сразу будет ложным.

Пример:

```
int main(void)
          // void можно
          // опустить, если используете
          // компилятор, который не поддерживает стандарт
          // C99 (см. Руководство
          // к вашему компилятору, возможно потребуется
          // установить опции соответствия требуемому
          // стандарту)
{
   int i,b;
   for (i=1; i<10; i++)
      b=i*i;

   return 0;
}
```

В этом примере вычисляются квадраты чисел от 1 до 9. (*Здесь* запись, например, **i++** эквивалентна записи **i = i+1**)

Некоторые варианты использования оператора **for** повышают его гибкость за счет возможности использования нескольких переменных, управляющих циклом.

Пример:

```
int main(void)
{
   int top,  bot;
   char arr[100],  temp;

// ...
// заполнение массива string данными
// ...

   for ( top=0, bot=99 ; top < bot ; top++, bot--)
   {
      temp=arr[top];
      arr[top]= arr[bot];
      arr[bot]=temp;
   }
   return 0;
 }
```

В этом примере, реализующем запись строки символов в обратном порядке, для управления циклом используются две переменные **top** и **bot**. Отметим, что на месте выражение 1 и выражение 3 здесь используются несколько выражений, записанных через запятую, и выполняемых последовательно.

Другим вариантом использования оператора **for** является бесконечный цикл. Для организации такого цикла можно использовать пустое условное выражение, а для выхода из цикла обычно используют дополнительное условие и оператор **break**.

Пример:

```
for (;;)
{ ...
  ...  break;
  ...
}
```

Так как согласно синтаксису языка **C** оператор может быть пустым, тело оператора **for** также может быть пустым. Такая форма оператора может быть использована для организации поиска.

Пример:

```
for (i=0; t[i]<10 ; i++) ;
```

В данном примере переменная цикла **i** принимает значение номера первого элемента массива **t**, значение которого больше 10.

• Оператор WHILE

Оператор цикла **while** называется циклом с предусловием и имеет следующий формат**:**

```
while (выражение) тело;
```

В качестве выражения допускается использовать любое выражение языка **C**, а в качестве тела любой оператор, в том числе пустой или составной. Схема выполнения оператора **while** следующая:

1. Вычисляется **выражение**.
2. Если выражение ложно, то выполнение оператора **while** заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполняется **тело** оператора **while**.
3. Процесс повторяется с пункта 1.

Оператор цикла вида

```
for (выражение-1; выражение-2; выражение-3) тело ;
```

может быть заменен оператором **while** следующим образом:

```
выражение-1;

while (выражение-2)
{   тело
    выражение-3;
}
```

Так же как и при выполнении оператора **for**, в операторе **while** вначале происходит проверка условия. Поэтому оператор **while** удобно использовать в ситуациях, когда тело оператора не всегда нужно выполнять.

Внутри операторов **for** и **while** можно использовать локальные переменные, которые должны быть объявлены с определением соответствующих типов.

- ## Оператор DO WHILE

Оператор цикла **do while** называется оператором цикла с постусловием и используется в тех случаях, когда необходимо выполнить тело цикла хотя бы один раз. Формат оператора имеет следующий вид:

```
do тело while (выражение);
```

Схема выполнения оператора do while :

1. Выполняется **тело** цикла (которое может быть составным оператором).
2. Вычисляется **выражение**.
3. Если выражение ложно, то выполнение оператора **do while** заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполнение оператора продолжается с пункта 1.

Чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор **break**.

Операторы **while** и **do while** могут быть вложенными.

Пример:

```
int i,j,k;
...
i=0;  j=0;  k=0;
do {
     i++;
     j--;
     while (a[k] < i)  k++;
    }
while (i<30 && j<-30);
```

- ## Оператор CONTINUE

Оператор **continue**, как и оператор **break**, используется только внутри операторов цикла, но в отличие от него выполнение программы продолжается не с оператора, следующего за прерванным оператором, а с начала прерванного оператора. Формат оператора следующий:

continue;

Пример:

```
int main(void)
{
   int a,b;
    for (a=1,b=0; a<100; b+=a,a++)
    {
        ...
           if (b%2) continue;
       ...    /* обработка четных сумм  */
    }
    return 0;
}
```

Когда сумма чисел **b** становится нечетной, оператор **continue** передает управление на очередную итерацию цикла **for**, не выполняя операторы обработки четных сумм.

Оператор **continue**, как и оператор **break**, относится к самому внутреннему из объемлющих его циклов.

- **УПРАЖНЕНИЯ**

Составить диаграммы видов деятельности (**Activity Diagram**) и программы на языке **C** для следующих заданий.

Упражнение 1

Реализовать задание, используя операторы цикла **for, while** и **do ... while** (т.е. всего **три** варианта программы).

Составить программу, которая работает следующим образом:

- спрашивает: сколько чисел хочет сложить пользователь
- запрашивает слагаемые (при помощи цикла)
- суммирует введенные данные
- выдает результат

Подсказка: **для суммы использовать вспомогательную переменную**.

Упражнение 2

Выбор. Реализовать задание, используя **if**
Составить программу, которая работает следующим образом:

- спрашивает у пользователя длину граней двух кубов
- вычисляет и выдает объем наибольшего куба (логично, что куб с наибольшей длиной грани будет больше)

Упражнение 3

Выбор. Реализовать задание, используя **switch**

Составить программу, которая работает следующим образом:

- спрашивает у пользователя два числа
- дает пользователю выбрать, хочет ли он эти числа сложить, из первого вычесть второе или умножить (т.е. имеется три варианта выбора)
- выполняет желаемое действие, выдает результат, информируя пользователя о произведенном действии.

Отчет оформляется в MS Word (или в другом текстовом редакторе), представляется в распечатанном виде. Отчет должен содержать необходимое для понимания описание работы. Представляемые программы должны быть прокомментированы.

Использованы материалы:

- ***Керниган Б., Ритчи Д. Язык С.***
- ***Громов, Титаренко***. Программирование на языке С.
- ***Материалы*** *Helena Kruus (MSc)*

Марина Брик

Составлено: 6.11.2007
Обновлено: 9.10.2008
Обновлено: 24.10.2010