Тема : знакомство с С (Си)

ОПЕРАЦИИ, for, while, do while, if, switch, упражнения

Содержание: операция = операции +,-,*,/,% комбинированные операции присваивания инкремент, декремент операции отношения логические операции операции сдвига поразрядные операции условная операция приоритеты операций циклы (упражнение 1,2 и 3) выбор (упражнения 4 и 5)

- ## Операции

Присваивание (=)

Оператор присваивания используется для присваивания значений переменным:

```
a = 10;
```

С помощью этой записи переменной **a** присваивается значение **10**. Необходимо обратить внимание, что правое значение присваивается левому.

Рассмотрим следующий пример:

```
a = b;
```

В этом случае переменная **a** получает значение переменной **b**. Необходимо понимать, что в этом случае – изменяется только значение переменной **a**, значение переменной **b** остается неизменным.

```
int a, b;  //значения переменных a и b не определены
a = 7;     //a присваиваивается значение 7, значения b еще нет
b = 5;     //a теперь равняется 7, b присваиваивается значение  5
a = b;     // a теперь равняется 5, b равняется 5
b = 13;    // a равняется 5, b теперь будет равным 13
```

Некоторые примеры использования:

- ```
  a = 5 + (b = 7);
  ```

  или

  ```
  b = 7;
  a = 5 + b;
  ```

- ```
  a = b = c = d = 6;
  ```

присвивается значение **6** переменным **d,c,b,a**

Арифметические операции

+	Сложение
–	Вычитание
*	Умножение
/	Деление
%	деление по модулю

Пояснения, возможно, из выше перечисленных операций требует только операция деления по модулю. Рассмотрим ее на примере:

```
a = 17%5;
```

Результатом операции является присвоение значения **2** переменной **a**, так как при делении чисел **17** и **5** получается частное **3** и **остаток 2** (то есть значение остатка и присваивается переменной **a**)

Комбинированные операции присваивания

Некоторые примеры:

- можем записать `a += b;` вместо записи `a = a+b;`
- `a /= 7;` тоже самое, что `a = a/7;`
- `a *= c + 9;` тоже самое, что `a = a*(c+9);`

Это правило действует и для других операторов арифметических операций.

Инкремент и декремент

Операция увеличения на единицу **(++)** и операция уменьшения на единицу **(--)**.

- Инкремент

```
a++;
a = a+1;
a += 1;
```

Все приведенные операции делают одно и тоже – увеличивают значение переменной **a** на 1.

Знак операции (**++**) может стоять перед переменной (**++a**) (префиксная форма записи) и после нее (**a++**) (постфиксная форма записи). Различие состоит в следующем:

`b = 5;`
`a = ++b;`
//значение **b** равняется 6
//**a** равняется 6
	`b = 5;`
	`a = b++;`
	//**b** равняется 5
	//значение **a** равняется 5, **b** равняется 6

В первом случае увеличивается значение **b**, и затем происходит присвоение значения **b** переменной **a**. Во втором случае значение переменной **b** присваивается переменной **a**, затем выполняется операция увеличения значения **b** .

- Декремент

```
a--;
a = a-1;
a -= 1;
```

Все приведенные операции делают одно и тоже – уменьшают значение переменной **a** на 1. Знак операции (**--**) может стоять перед переменной (**--a**) и после нее (**a--**). Здесь мы сталкиваемся с ситуацией, описанной выше (см. инкремент).

Операции отношения

- **(a == b)** – **равно** используется, когда надо сравнить значения **a** и **b** на равенство (**NB!** Надо обратить внимание на то, что здесь используется знак = =. Выражение **a = b** обозначает в С присваивание!)
- **(a > b)** – **больше**
- **(a < b)** – **меньше**
- **(a != b)** – **не равно**
- **(a >= b)** – **больше или равно**
- **(a <= b)** – **меньше или равно**

Логические операции

- **&&** - логическое **И (AND)**
- **||** - логическое **ИЛИ (OR)**

К логическим операциям относятся операция логического И (&&) и операция логического ИЛИ (||). Операнды логических операций могут быть целого типа, плавающего типа или типа указателя, при этом в каждой операции могут участвовать операнды различных типов.

Операнды логических выражений вычисляются слева направо. Если значения первого операнда достаточно, чтобы определить результат операции, то второй операнд не вычисляется.

Логические операции не вызывают стандартных арифметических преобразований. Они оценивают каждый операнд с точки зрения его эквивалентности нулю. Результатом логической операции является 0 или 1, тип результата int.

Операция логического И (&&) вырабатывает значение 1, если оба операнда имеют ненулевые значения. Если один из операндов равен 0, то результат также равен 0. Если значение первого операнда равно 0, то второй операнд не вычисляется.

Операция логического ИЛИ (||) выполняет над операндами операцию включающего ИЛИ. Она вырабатывает значение 0, если оба операнда имеют значение 0, если какой-либо из операндов имеет ненулевое значение, то результат операции равен 1. Если первый операнд имеет ненулевое значение, то второй операнд не вычисляется.

Операции сдвига

Операции сдвига осуществляют смещение операнда влево (<<) или вправо (>>) на число битов, задаваемое вторым операндом. Оба операнда должны быть целыми величинами.

Выполняются обычные арифметические преобразования. При сдвиге влево правые освобождающиеся биты устанавливаются в нуль. При сдвиге вправо метод заполнения освобождающихся левых битов зависит от типа первого операнда. Если тип unsigned, то свободные левые биты устанавливаются в нуль. В противном случае они заполняются копией знакового бита. Результат операции сдвига не определен, если второй операнд отрицательный.

Примеры:

```
int  i=0x1234,   k ;  /*       i= 0001 0010 0011 0100   */

 k = i<<8 ;           /*       j = 0x3400   */
```

Поразрядные операции

- `&` – конъюнкция (И)
- `|` – дизъюнкция (ИЛИ)
- `^` – исключающее ИЛИ (XOR)

К поразрядным операциям относятся: операция поразрядного логического "И" (&), операция поразрядного логического "ИЛИ" (|), операция поразрядного "исключающего ИЛИ" (^).

Операнды поразрядных операций могут быть любого целого типа. При необходимости над операндами выполняются преобразования по умолчанию, тип результата - это тип операндов после преобразования.

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.

Пример:

```
int  i=0x45FF,       /*       i= 0100 0101 1111 1111   */
     j=0x00FF;       /*       j= 0000 0000 1111 1111   */
char r;
      r = i^j;     /* r=0x4500 = 0100 0101 0000 0000   */
      r = i|j;     /* r=0x45FF = 0100 0101 1111 1111   */
      r = i&j;     /* r=0x00FF = 0000 0000 1111 1111 */
```

дополнительная информация:

The GNU C Programming Tutorial - Bitwise operators
http://crasseux.com/books/ctutorial/Bitwise-operators.html#Bitwise%20operators
[26.10.2010]

Условная операция

В языке **C** имеется одна тернарная операция - условная операция, которая имеет следующий формат:

```
операнд-1 ? операнд-2 : операнд-3
```

Операнд-1 должен быть целого или плавающего типа или быть указателем. Он оценивается с точки зрения его эквивалентности 0. Если **операнд-1** не равен 0, то вычисляется **операнд-2** и его значение является результатом операции. Если **операнд-1** равен 0, то вычисляется **операнд-3** и его значение является результатом операции. Следует отметить, что вычисляется либо операнд-2, либо операнд-3, но не оба. Тип результата зависит от типов операнда-2 и операнда-3, следующим образом.

Пример:

```
a>b ? a : b
```

Возвращает **a** в случае, если **a** больше **b** и возвращает **b** в случае, если **a** не больше чем **b.**

Функциональность аналогичная оператору выбора **if**, различие состоит в том, что условную операцию можно использовать для присвоения значений переменным и даже внутри выражений.

Пример:

```
max = (d<=b) ? b : d;
```

Переменной max присваивается максимальное значение переменных d и b.

Приоритеты операций и порядок вычислений

В языке **C** операции с высшими приоритетами вычисляются первыми. Наивысшим приоритетом является приоритет равный 1. Приоритеты и порядок операций приведены в табл.1.

Дополнительную информацию о приоритете операций см., например:

The GNU C Programming Tutorial - Precedence of operators

http://crasseux.com/books/ctutorial/Precedence-of-operators.html [26.10.2010]

Таблица 1

Приоритет	Знак операции	Типы операции	Порядок выполнения
1	- ~ ! * & ++ -- sizeof приведение типов	Унарные	Справа налево
2	() [] . ->	Выражение	Слева направо
3	* / %	Мультипликативные	
4	+ -	Аддитивные	
5	<< >>	Сдвиг	
6	< > <= >=	Отношение	
7	== !=	Отношение (равенство)	
8	&	Поразрядное И	Слева направо
9	^	Поразрядное исключающее ИЛИ	
10	|	Поразрядное ИЛИ	
11	&&	Логическое И	
12	||	Логическое ИЛИ	
13	? :	Условная	
14	= *= /= %= += -= &= |= >>= <<= ^=	Простое и составное присваивание	Справа налево
15	,	Последовательное вычисление	Слева направо

- **Циклы**

  - **FOR**

(источник: Teodor Luczkowski. Baasteadmised programmeerimiskeelest C++)

Следующий for-цикл выдает ряд ”a8 b7 c6 d5 e4 f3 g2 h1”:

```
for (i=8, c='a'; i>0;printf("%c%d ",c++, i--));
```

- **Упражнение 1**

Найти в тексте программы ошибки и заставить программу работать как ожидается. Использовать операцию инкремента.

Планируемый ход работы программы:

```
Sisesta täisarv: 5
number: 1
number: 2
number: 3
```

```
number: 4
number: 5

#include <stdio.h>
int main (void)
{
        int i, arv;

 //ввод целого числа
    printf("Sisesta täisarv: ");
    scanf("%d",arv);

 //цикл, в ходе которого выдаются числа, начиная с 1 и до
 //введенного пользователем значения
    for ( i=1; i = arv; i=i+1 )
    {
              printf ("number: %d\n",i);
    }

    getchar();
    return 0
}
```

- **WHILE**

- **Упражнение 2**

Найти в тексте программы ошибки и заставить программу работать как ожидается.

Планируемый ход работы программы:

```
Sisesta täisarv: 5
number: 1
number: 2
number: 3
number: 4
number: 5

int main (void)
{

    int i, arv;

    printf("Sisesta täisarv: ");
    scanf("%d",&arv);

    i=1;

  //цикл, в ходе которого выдаются числа, начиная с 1 и до
  //введенного пользователем значения
```

```
    while (i<arv )
    {
            printf("number: %d\n",i);
            i++;
    }

    getchar();
    return 0;
}
```

- **DO WHILE**

Пример

```
#include <stdio.h>
int main (void)
{
    int n;

  //цикл повторяется до тех пор, пока пользователь не введет 0
    do
    {
            printf("\nSisesta 0 tsykli lõpetamiseks: ",n);
            scanf("%d",&n);
            printf("Sa sisestasid täisarvu: %d\n",n);
    } while (n!=0);

    getchar();
    return 0;
}
```

- **Упражнение 3**

Найти в тексте программы ошибки и заставить программу работать как ожидается.

Планируемый ход работы программы:

```
Sisesta täisarv: 5
number: 1
number: 2
number: 3
number: 4
number: 5
```

```
int main (void)
{
    int i;

    printf("Sisesta täisarv: ");
    scanf("%d",&arv);
```

```
    do
    {
             printf("number: %d\n",i);
             i++;
    } while (i<arv)

    getchar();
    return 0;
}
```

- **Выборы (см. Занятие 4,5)**

  - **IF**

Пример

```
#include <stdio.h>

int main (void)
{
      int arv;

    printf("Sisesta taisarv: ");
    scanf("%d",&arv);

    if(arv != 100)
    {
           printf("Sa ei sisestanud 100!!! :( \n");
           printf("Sa sisestasid %d\n", arv);
    }

    getchar();
    return 0;
}
```

- **Упражнение 4**

Найти в тексте программы ошибки и заставить программу работать как ожидается:

Если пользователь вводит 100, то на терминал выдается `Sa sisestasid 100!!! :)`**, в противном случае:** `Sa ei sisestanud 100... :(`

```
  #include <stdio.h>

int main (void)
{
    int arv;

    printf("Sisesta taisarv: ");
```

```
    scanf("%d",arv);

    if(arv != 100)
    {
           printf("Sa sisestasid 100!!! :) \n")
    }
    else
    {
           printf("Sa ei sisestanud 100... :( \n");
    }


    getchar();
    return 0;
}
```

- **SWITCH**

- **Упражнение 5**

Найти в тексте программы ошибки и заставить программу работать как ожидается:

Пользователь вводит цифру, соответствующую своему выбору. Печатается соответствующая выбору информация.

```
#include <stdio.h>

int main (void)
{
        int valik;

  //создание меню
    printf("Tee oma valik\n\n");
    printf("1 - PRINTIMINE\n");
    printf("2 - FAILI SALVESTAMINE\n");
    printf("3 - ANDMETE EKSPORTIMINE\n");
    printf("0 - TÖÖ LÕPETAMINE\n\n");

    scanf("%d",&valik);

    //выбор значения
    switch(valik){
       case 1: printf("Valisid PRINTIMISE");
       case 2: printf("Valisid FAILI SALVESTAMISE");
       case 3: printf("Valisid ANDMETE EKSPORTIMISE");
       case 0: printf("Valisid TÖÖ LÕPETAMISE");
       default: printf("VIGA!!! Sellist valikut ei ole");
    }

    getchar();
    return 0;
```

```
}
```

Отчет оформляется в MS Word (или в другом текстовом редакторе), представляется в распечатанном виде. Найденные ошибки должны быть выделены и описаны.

Использованы материалы:

- ***Громов, Титаренко***. Программирование на языке С.
- ***Teodor Luczkowski.*** Baasteadmised programmeerimiskeelest C++
- ***Материалы*** *Helena Kruus (MSc)*

Марина Брик

Составлено: 15.11.2007
Обновлено: 9.10.2008
26.10.2010