Оператор-выражение, условный оператор и условная операция, составной оператор, оператор switch. (ОК)

План

1. Оператор-выражение
2. Условный оператор
3. условная операция
4. Составной оператор
5. Оператор switch

1. Опрератор-выражение

Выражения формируют основные строительные блоки для операторов и определяют, каким образом программа управляет данными и изменяет их.

Операторы определяют каким образом управление передается из одной части программы другой.

В языке Си любое выражение можно «превратить» в оператор, добавив к этому выражению точку с запятой: ++i;

В языке Си точка с запятой является элементом оператора и его завершающей частью, а не разделителем операторов. i=1;

1 сохраняется в переменной i, затем значение операции (новое значение переменной i) вычисляется, но не используется.

```
i * j – 1;      // warning: statement with no effect
```

Поскольку переменные i и j не изменяются, этот оператор не имеет никакого эффекта и бесполезен.

2.Условный оператор if-else

Условный оператор позволяет сделать выбор между двумя альтернативами, проверив значение выражения.

```
if (выражение)
    оператор_1
else
    оператор_2
```

Скобки вокруг выражения обязательны, они являются частью самого условного оператора. Часть else не является обязательной.

Поскольку часть else условного оператора может отсутствовать, в случае вложенных условных операторов это может приводить к путанице.

В языке Си else всегда связывается с ближайшим предыдущим оператором if без else.

```
if (y != 0)
    if (x != 0)
        result = x / y;
    else
        printf(“y is equal to 0\n”);
```

Поскольку в выражении условного оператора анализируется числовое значение этого выражения, отдельные конструкции можно упростить.

```
if (выражение != 0)    <=>   if (выражение)
```

3. Условная операция

Условная операция состоит из двух символов «?» и «:», которые используются вместе следующим образом

```
expr_1 ? expr_2 : expr_3
```

Сначала вычисляется выражение expr_1.

Если оно отлично от нуля, то вычисляется выражение expr_2, и его значение становится значением условной операции.

Если значение выражение expr_1 равно нулю, то значением условной операции становится значение выражения expr_3.

4.Составной оператор

Необходимо использовать составной оператор.

Заключая несколько операторов в фигурные скобки, мы заставляем компилятор интерпретировать их как один оператор.

```
{
        оператор1
```

```
            оператор2
        }
```

5. Оператор switch

В общей форме оператор switch может быть записан следующим образом

```
switch (выражение)
{
    case констнатное_выражение : операторы
    ...
    case констнатное_выражение : операторы
    default : операторы
}
```

Управляющее выражение, которое располагается за ключевым словом switch, обязательно должно быть целочисленным

Константное выражение – это обычное выражение, но оно не может содержать переменных и вызовов функций.

5 константное выражение
5 + 10 константное выражение
n + 10 НЕ константное выражение

После каждого блока case может располагаться любое число операторов. Никакие скобки не требуются.

Последним оператором в группе таких операторов обычно бывает оператор break.

Только одно константное выражение может располагаться в case-метке.

Но несколько case-меток могут предшествовать одной и той же группе операторов.

```
   switch (mark)
   {
      case 5:
      case 4:
      case 3:  printf("Passing\n");
               break;
      case 2:  printf("Failing\n");
               break;
      default: printf("Illegal mark\n");
               break;
  }
```

case-метки не могут быть одинаковыми.

Порядок case-меток (даже метки default) не важен.

case-метка default не является обязательной.