Перечисления

Перечисление – это набор именованных целых констант. Перечисления довольно часто встречаются в повседневной жизни. Вот, например, перечисление, в котором приведены названия монет, используемых в Соединенных Штатах:

penny (пенни, монета в один цент), nickel (никель, монета в пять центов), dime (монета в 10 центов), quarter (25 центов, четверть доллара), half-dollar (полдоллара), dollar (доллар)

Перечисления определяются во многом так же, как и структуры; началом объявления перечислимого типа[1] служит ключевое слово `enum`. Перечисление в общем виде выглядит так:

```
enum тег {список перечисления} список переменных;
```

Здесь *тег* и *список переменных* не являются обязательными. (Но хотя бы что-то одно из них должно присутствовать.) Следующий фрагмент кода определяет перечисление с именем `coin` (монета):

```
enum coin { penny, nickel, dime, quarter,
            half_dollar, dollar};
```

Тег перечисления можно использовать для объявления переменных данного перечислимого типа. Вот код, в котором `money` (деньги) объявляется в качестве переменной типа `coin`:

```
enum coin money;
```

С учетом этих объявлений совершенно верными являются следующие операторы:

```
money = dime;
if(money==quarter) printf("Денег всего четверть доллара.\n");
```

Главное, что нужно знать для понимания перечислений – каждый их элемент[2] представляет целое число. В таком виде элементы перечислений можно применять везде, где используются целые числа. Каждому элементу дается значение, на единицу большее, чем у его предшественника. Первый элемент перечисления имеет значение 0. Поэтому, при выполнении кода

```
printf("%d %d", penny, dime);
```

на экран будет выведено `0 2`.

Однако для одного или более элементов можно указать значение, используемое как инициализатор. Для этого после перечислителя надо поставить знак равенства, а затем – целое значение. Перечислителям, которые идут после инициализатора, присваиваются значения, большие предшествующего. Например, следующий код присваивает `quarter` значение 100:

```
enum coin { penny, nickel, dime, quarter=100,
            half_dollar, dollar};
```

вот какие значения появились у этих элементов:

```
penny           0
nickel          1
dime            2
quarter         100
half_dollar     101
dollar          102
```

Относительно перечислений есть одно распространенное, но ошибочное мнение. Оно состоит в том, что их элементы можно непосредственно вводить и выводить. Это не так. Например, следующий фрагмент кода не будет выполняться так, как того ожидают многие неопытные программисты:

```
/* этот код работать не будет */
money = dollar;
printf("%s", money);
```

Здесь dollar – это имя для значения целого типа; это не строка. Таким образом, попытка вывести money в виде строки по существу обречена. По той же причине для достижения нужных результатов не годится и такой код:

```
/* этот код не правильный */
strcpy(money, "dime");
```

То есть строка, содержащая имя элемента, автоматически в этот перечислитель не превратится.

На самом же деле создавать код для ввода и вывода элементов перечислений – это довольно-таки скучное занятие (но его можно избежать лишь тогда, когда будет достаточно именно целых значений этих перечислителей). Например, чтобы выводить название монеты, вид которой находится в money, потребуется следующий код:

```
switch(money) {
  case penny: printf("пенни");
    break;
  case nickel: printf("никель");
    break;
  case dime: printf("монета в 10 центов");
    break;
  case quarter: printf("четверть доллара");
    break;
  case half_dollar: printf("полдоллара");
    break;
  case dollar: printf("доллар");
}
```

Иногда можно объявить строчный массив и использовать значение перечисления как индекс при переводе этого значения в соответствующую строку. Например, следующий код также выводит нужную строку:

```
char name[][12]={
  "пенни",
  "никель",
  "монета в 10 центов",
  "четверть доллара",
  "полдоллара",
  "доллар"
};
printf("%s", name[money]);
```

Конечно, он будет работать только тогда, когда не инициализирован ни один из элементов перечисления, так как строчный массив должен иметь индекс, который начинается с 0 и возрастает каждый раз на 1.

Так как при операциях ввода/вывода необходимо специально заботиться о преобразовании перечислений в их строчный эквивалент, который можно легко прочитать, то перечисления полезнее всего именно в тех процедурах, где такие преобразования не нужны. Например, перечисления часто применяются, чтобы определить таблицы соответствия символов в компиляторах.

---

[1] Иногда используется термин *перечисляемый тип*.

[2] Элементы списка перечисления называются также *перечислителями* и *идентификаторами*.

Содержание | <<< | >>>
[Главная] [Гостевая]