Указатель представляет собой переменную, хранящую адрес какого-нибудь другого

объекта, например другой переменной. Так, если в переменной х хранится адрес переменной

у, то говорят, что переменная х указывает на переменную у. Когда указатель

указывает на переменную, то значение этой переменной может быть получено или

изменено по указателю. Такие операции с указателями называют *непрямой адресацией.*

**Объявление указателя.**

Переменные-указатели должны быть объявлены как таковые. Ниже приведена общая

форма объявления переменной-указателя:

***тип\* имя\_переменной;***

где *тип* обозначает *соотносимый тип,* который не должен быть ссылочным. Это означает,

что в C# нельзя объявить указатель на объект определенного класса. Соотносимый

тип указателя иногда еще называют *базовым.* Обратите внимание на положение

знака \* в объявлении указателя. Он должен следовать после наименования типа.

А *имя\_переменной* обозначает конкретное имя указателя-переменной.

Следует, однако, иметь в виду, что указателю ничто не мешает указывать на что угодно.

Именно поэтому указатели потенциально небезопасны.

Если у вас есть опыт программирования на С/С++, то вы должны ясно понимать

главное отличие в объявлении указателей в C# и C/C++. При объявлении указателя

в C/C++ знак \* *не* разделяет список переменных в объявлении. Поэтому в следующей

строке кода:

***int\* р, q;***

объявляется указатель р типа int и переменная q типа int. Это равнозначно двум

следующим объявлениям.

***int\* р;***

***int q;***

А в C# знак \* *является* разделительным, и поэтому в объявлении

***int\* р, q;***

создаются две переменные-указателя. Это равнозначно двум следующим объявлениям.

***int\* р;***

***int\* q;***

Это главное отличие следует иметь в виду при переносе кода C/C++ на С#.

**Операторы \* и & в указателях**

В указателях применяются два оператора: \* и &. Оператор & является унарным

и возвращает адрес памяти своего операнда.

int\* ip;

int num = 10;

ip = &num;

в переменной ip сохраняется адрес памяти переменной num. Это адрес расположения

переменной num в оперативной памяти компьютера. Он не имеет *никакого* отношения

к *значению* переменной num. Поэтому в переменной ip содержится не значение

10, являющееся исходным для переменной num, а конкретный адрес, по которому эта

переменная хранится в оперативной памяти. Операцию & можно рассматривать как

возврат адреса той переменной, перед которой она указывается. Таким образом, приведенное

выше присваивание словами можно описать так: "Переменная ip получает

адрес переменной num."

Второй оператор, \*, является дополнением оператора &. Этот унарный оператор находит

значение переменной, расположенной по адресу, на который указывает его операнд.

Следовательно, этот оператор обращается к значению переменной, на которую

указывает соответствующий указатель. Так, если переменная ip содержит адрес памяти

переменной num, как показано в предыдущем примере, то в следующей строке кода:

int val = \*ip;

в переменной val сохраняется значение 10 переменной num, на которую указывает

переменная ip. Операцию \* можно рассматривать как получение значения по адресу.

Поэтому приведенный выше оператор присваивания описывается словами следующим

образом: *"*Переменная val получает значение по адресу, хранящемуся в переменной

ip.