**Указатель void \***

В C++ существует специальный тип указателя, который называется указателем на неопределённый тип. Для определения такого указателя вместо имени типа используется ключевое слово void в сочетании с описателем, перед которым располагается символ ptrОперации \*.

void \*UndefPoint;

С одной стороны, объявленная подобным образом переменная также является объектом определённого типа - типа указатель на объект неопределённого типа. В Borland C++ 4.5 имя UndefPoint действительно ссылается на объект размером в 32 бита со структурой, которая позволяет сохранять адреса.

Но, с другой стороны, для объекта типа указатель на объект неопределённого типа отсутствует информация о размерах и внутренней структуре адресуемого участка памяти. Из-за этого не могут быть определены какие-либо операции для преобразования значений.

Поэтому переменной UndefPoint невозможно присвоить никаких значений без явного преобразования этих значений к определённому типу указателя.

UndefPoint = 0xb8000000; // Такое присвоение недопустимо.

Подобный запрет является вынужденной мерой предосторожности. Если разрешить такое присвоение, то неизвестно, как поступать в случае, когда потребуется изменить значение переменной UndefPoint, например, с помощью операции инкрементации.

UndefPoint++; // Для типа void \* нет такой операции…

Эта операция (как и любая другая для типа указатель на объект неопределённого типа) не определена. И для того, чтобы не разбираться со всеми операциями по отдельности, лучше пресечь подобные недоразумения "в корне", то есть на стадии присвоения значения.

Объектам типа указатель на объект неопределённого типа в качестве значений разрешается присваивать значения лишь в сочетании с операцией явного преобразования типа.

В этом случае указатель на объект неопределённого типа становится обычным указателем на объект какого-либо конкретного типа. Со всеми вытекающими отсюда последствиями.

Но и тогда надо постоянно напоминать транслятору о том типе данных, который в данный момент представляется указателем на объект неопределённого типа:

int mmm = 10;

pUndefPointer = (int \*)&mmm;

pUndefPointer выступает в роли указателя на объект типа int.

(\*(int \*)pUndefPointer)++;

Для указателя на объект неопределённого типа не существует способа непосредственной перенастройки указателя на следующий объект с помощью операции инкрементации. В операторе, реализующем операции инкрементации и декрементации, только с помощью операций явного преобразования типа можно сообщить транслятору величину, на которую требуется изменить первоначальное значение указателя.

pUndefPointer++; // Это неверно, инкрементация не определена…

(int \*)pUndefPointer++; // И так тоже ничего не получается…

((int \*)pUndefPointer)++; // А так хорошо… Сколько скобок!

++(int \*)pUndefPointer; // И вот так тоже хорошо…

С помощью операции разыменования и с дополнительной операцией явного преобразования типа изменили значение переменной mmm.

pUndefPointer = (int \*)pUndefPointer + sizeof(int);

Теперь перенастроили указатель на следующий объект типа int.

pUndefPointer = (int \*)pUndefPointer + 1;

И получаем тот же самый результат.

Специфика указателя на объект неопределённого типа позволяет выполнять достаточно нетривиальные преобразования:

(\*(char \*)pUndefPointer)++;

А как изменится значение переменной mmm в этом случае?

pUndefPointer = (char \*)pUndefPointer + 1;

Указатель перенастроился на объект типа char. То есть просто сдвинулся на 1байт.

Работа с указателями на объекты определённого типа не требует такого педантичного напоминания о типе объектов, на которые настроен указатель. Транслятор об этом не забывает.

int \* pInt;

int mmm = 10;

pInt = &mmm; // Настроили указатель.

pInt++; // Перешли к очередному объекту.

\*pInt++; // Изменили значение объекта, идущего следом за

// переменной mmm.

Напомним, что происходит в ходе выполнения этого оператора.

* после выполнения операции разыменования вычисляется значение (адрес объекта mmm),
* это значение становится значением выражения,
* после чего это значение увеличивается на величину, кратную размеру того типа данного, для которого был объявлен указатель.

Операции явного преобразования типов позволяют присваивать указателям в качестве значений адреса объектов типов, отличных от того типа объектов, для которого был объявлен указатель:

int mmm = 10;

char ccc = 'X';

float fff = 123.45;

pInt = &mmm;

pNullInt = (int \*)&ccc;

pNullInt = (int \*)&fff; // Здесь будет выдано предупреждение об

// опасном преобразовании.

Это обстоятельство имеет определённые последствия, которые связаны с тем, что все преобразования над значениями указателей будут производиться без учёта особенностей структуры тех объектов, на которые указатель в самом начале был настроен.

При этом ответственность за результаты подобных преобразований возлагается на программиста.