Косвенный доступ к данным

Переменные, с которыми вы имели дело до сих пор, обеспечивали вам возможность именовать некоторое место в памяти, в котором можно сохранять данные определенного типа. Содержимое переменной либо вводилось из некоторого внешнего источника, такого как клавиатура, либо вычислялось на основе других введенных значений. Но есть и другой вид переменных С++, не хранящих в себе данные, которые вы обычно используете для вычислений, но значительно повышающих мощь и гибкость ваших программ. Переменные такого рода называются *указателями*.

Что такое указатель?

Каждое место в памяти, которое вы используете для хранения данных, имеет адрес. Адрес предоставляет возможность для оборудования компьютера обращаться к определенному элементу данных. Строго говоря, это только вы обращаетесь к переменным по имени, а внутри «железяки» - по адресам. Но, как выяснилось, и нам это не помешает.

Указатель - это переменная, которая сохраняет адрес другой переменной определенного типа.

Таким образом, если мы хотим обратиться к данным по адресу, этот адрес нужно сохранить, а храниться он в переменных определенного вида – указателях.

И в зависимости от кого , какого типа данные будут храниться по этому адресу, будут определяться сами указатели.

Переменная - указатель обладает именем, как и любая другая переменная, и также имеет тип, определяющий то, на какого рода данные она может указывать. Переменная, являющаяся указателем, содержащим адрес местоположения в памяти, которое хранит значение типа *int*, называется «указателем на *int*».

Объявление указателей

Объявления указателей подобно объявлению обычных переменных, за исключением того, что имя дополняется впереди звездочкой, говорящей о том, что данная переменная является указателем.

*тип\* имя указателя на переменную заданного типа*

Например, чтобы объявить указатель *pnumber* типа *long*, можно использовать следующий оператор:

*long \* pnumber;*

А для объявления указателя *pCount*, в котором будут храниться адреса переменных типа *int* нужно поступить таким образом:

*int \* pCount;*

Эти объявления записаны со звездочкой, расположенной ближе к имени переменной. Если хотите, можете также написать так:

*long\* pnumber;*

Компилятору это безразлично. Однако типом переменной *pnumber* является "указатель на long", что часто обозначается помещением звездочки ближе к имени типа. Не важно, какой стиль вы предпочтете- важно придерживаться его согласованно.

Можно смешивать объявления обычных переменных и указателей в одном операторе, например:

*long\* pnumber, number = 99;*

Этот оператор, как и ранее, объявляет указатель *pnumber* типа "указатель на long", но также объявляет переменную *number* типа *long*. В конечном итоге, вероятно, лучше все-таки объявлять указатели отдельно от других переменных; в противном случае такой оператор может ввести в заблуждение в отношении типа объявляемых переменных, особенно если вы предпочитаете размещать \* рядом с именем типа. Следующие операторы, безусловно, выглядят яснее, к тому же размещение объявления в отдельных строках позволяет вам добавить им индивидуальные комментарии, что облегчает чтение программы.

*long number =99; // Объявление и инициализация переменной long*

*long\* pnumber; // Объявление переменной типа указатель на long*

В С++ существует общепринятое соглашение- называть указатели именами, начинающимися с буквы *р*. Это позволяет сразу увидеть, какие переменные явля­ются указателями, что, в свою очередь, упрощает восприятие программ.

Предположим, что у вас есть целочисленная переменная типа *int* по имени *number*, объявленная выше и содержащая значение 99. Кроме того, у вас есть указатель на *int* по имени *pnumber*, который вы можете использовать для хранения адреса переменной *number*. Но как получить адрес этой переменной?

Операция получения адреса

Что вам нужно для этого – операция получения адреса, &. Она представляет со­бой унарную операцию, возвращающую адрес переменной - своего операнда. Чтобы присвоить значение адреса переменной указателю, вы можете написать следующий оператор присваивания:

*pnumber =&number; // Сохранить адрес number в pnumber*

Результат выполнения этой операции можно видеть на рисунке.



Вы можете использовать операцию & для получения адреса любой переменной, но для сохранения его вам понадобится переменная-указатель соответствующего типа. Например, если нужно сохранить адрес переменной *double*, указатель должен быть объявлен с типом *double* \*, то есть "указатель на *double*" .

Использование указателей

Получение адреса переменной и сохранение его в переменной-указателе- это очень хорошо, но более интересно, как его использовать. Основной смысл приме­нения указателя состоит в возможности доступа к данным, на которые он указывает. Это делается с помощью операции разыменования (\*).

Операция разыменования

Для доступа к переменной, на которую указывает указатель, вы будете применять операцию разыменования указателя (\*), подставляя в качестве операнда имя пере­менной-указателя. Название «операция разыменования» или «операция косвенного доступа» говорит о том, что обращение к данным не прямое, и что для получения доступа к данным, на которые указывает указатель, его следует «разыменовать».

Один аспект этой операции, который может показаться запутанным, связан с существованием разных применений одного и того же символа \*. Во-первых, это- операция арифметического умножения, во-вторых, операция разыменования, и вдобавок она используется при объявлении переменных-указателей. Всякий раз, встречая символ \*, компилятор определяет его значение в каждом конкретном случае из контекста. Когда вы перемножаете две переменных, например, *А\*В*, не существует никакой другой осмысленной интерпретации этого выражения, кроме как операции умножения

*int count1 = 1;*

*int count2= 1;*

*int\* pcount;*

*pcount = &count1;*

*count2 = \*pcount \* 10;*

**Зачем нужны указатели?**

Вопрос, который обычно возникает при изучении указателей, звучит так: "Зачем они вообще нужны?". В конце концов, получить адрес переменной, которая вам уже известна, сохранить его в указателе, чтобы потом разыменовать, выглядит излишним, то есть чем-то таким, без чего вполне можно обойтись. Однако существует несколько причин, объясняющих важность указателей.

Как вы вскоре увидите, нотацию указателей можно использовать для операций с данными, хранимыми в массиве; иногда это выполняется быстрее, чем в случае применения обычной нотации массивов. К тому же, когда позднее вы будете определять свои собственные функции, то увидите, что указатели интенсивно используются для обеспечения доступа функций к объемным блокам данных, таким как массивы, которые определены вне этих функций. Но как вы увидите позднее, еще более важно то, что память для переменных может выделяться динамически, то есть во время выполнения программы. Это средство позволяет программам адаптировать использование памяти к объему обрабатываемых данных. Поскольку при этом вы не можете знать заранее, сколько переменных придется создать динамически, основной способ работы с такой памятью - через указатели. Так что, будьте уверены, что указатели вам пригодятся.

Инициализация указателей

Использование неинициализированных указателей чрезвычайно опасно. Вы можете легко перезаписать произвольную область памяти через неинициализированный указатель. Полученный ущерб при этом зависит лишь от степени вашего везения потому инициализировать указатели - более чем хорошая идея.

Очень легко инициализировать указатель адресом переменной, которая уже определена. Можно инициализировать указатель *pnumber*адресом переменной *number*, просто применив операцию & к имени переменной:

*pnumber = &numberl; // Сохранить адрес в указателе*

\**pnumber +=11*; // *Увеличить number на 11*

Инициализируя указатель адресом другой переменной, помните, что эта переменная должна быть объявлена до объявления указателя.

Конечно, объявляя указатель, вы можете решить не инициализировать его адресом определенной переменной. В этом случае его можно инициализировать указателем, эквивалентным нулю. Для этого в Visual С++ предусмотрен символ *NULL*, который уже определен как 0, поэтому указатель можно объявлять и инициализировать с помощью следующего оператора вместо того, что вы видели ранее:

*int\* pnumber = NULL; // Указатель, не указывающий ни на что*

Это гарантирует, что указатель не содержит адреса, который воспринимается, как корректный, и представляет ему значение, которое можно проверять в операторе *if*, вроде такого:

*if (pnumber == NULL)*

*cout << endl << "pnumber is null.";*

Конечно, вы можете инициализировать указатель явно числом 0, что также га­рантирует значение, не указывающее ни на что. Ни один объект не может быть раз­мещен по адресу 0, поэтому 0 применяется, как адрес, означающий, что у указателя нет цели. К тому же, если вы собираетесь собирать свой код другими компиляторами, лучше использовать 0 для инициализации нулевых указателей.

Для использования 0 в качестве инициализирующего значения указателя, нужно просто написать:

*int\* pnumber = 0; // Указатель, не указывающий ни на что*

Чтобы проверить, содержит ли указатель корректный адрес, используйте следующий оператор:

*if (pnumber == 0)*

*cout << endl << "pnumber equal null.";*

С тем же успехом можете применить и такой оператор:

*if(!pnumber)*

*cout << endl << "pnumber equal null.";*

Адрес, на который указывает NULL указатель, содержит «мусорное» значение. Нельзя разыменовывать нулевой указатель, что приведет к немедленному прерыванию программы.

Указатели и массивы.

Имена массивов в некоторых случаях могут вести себя как указатели. В большинстве ситуаций, если вы используете имя одномерного массива само по себе, оно автоматически преобразуется в указатель на первый элемент этого массива.

Если имеются следующие объявления:

*double\* pdata;*

*double data[5];*

то вы можете написать такое присваивание:

*pdata = data; // Инициализация указателя адресом массива*

Происходит присваивание адреса первого элемента массива *data* указателю *pdata*. Применение имени массива самого по себе означает ссылку на его адрес. Если вы используете имя массива data с индексным значением, то это означает ссылку на содержимое элемента, соответствующего значению индекса. Поэтому, если вы хотите сохранить адрес элемента в указателе, то должны использовать операцию взятия адреса:

*pdata = &data[1];*

Здесь указатель pdata получает адрес второго элемента массива.

Арифметика указателей

Над указателями можно выполнять арифметические операции. Правда, они ограничены только сложением и вычитанием, но можно также сравнивать значения указателей, получая логический результат. Арифметика над указателями неявно предполагает, что указатель указывает на массив, и арифметические операции выполняются над адресом, содержащимся в указателе. Так, например, указателю *pdata* можно присвоить адрес третьего элемента массива *data* с помощью следующего оператора:

*pdata = &data[2];*

В этом случае выражение *pdata + 1* будет ссылаться на адрес *data [ 3 ]* - четвертого элемента массива *data*, поэтому вы можете переставить указатель на этот элемент следующим образом:

*pdata +=1; // Инкремент указателя pdata переносит его на //следующий элемент*

Этот оператор увеличивает адрес, содержащийся в pdata, на количество байт, которое занимает каждый элемент массива data.



Другими словами, инкремент и декремент указателя работает в терминах типа объекта, на который он указывает. Увеличение на единицу указателя на *long*изменяет его содержимое на адрес следующего *long*, то есть увеличивает его адрес на восемь. Аналогично, инкремент указателя на *short* на единицу увеличивает значение адреса на два. Более распространенная нотация для увеличения указателя использует операцию инкремента.

Например:

*pdata++; // Увеличить pdata до следующего элемента*

Это эквивалентно форме +=, к тому же более часто применяется. Однако, использование формы +=, подчеркивает, что хотя обычно значение инкремента равно единице, эффект от его применения к указателю выражается в увеличении адреса больше чем на единицу, за исключением случая указателя на *char*.

Адрес, полученный в результате применения арифметической операции к указателю, может изменяться от адреса первого элемента массива до адреса, лежащего сразу за его последним элементом. Вне этих пределов поведение указателя не определено.

Можно, конечно, разыменовать указатель, к которому применено арифметическое действие (а иначе в нем не было бы особого смысла).

Например, если предположить, что *pdata* все еще указывает на *data [2]*, то оператор:

*\* (pdata + 1) = \* (pdata + 2) ;*

эквивалентен следующему:

*data[3] = data[4];*

Когда вы хотите разыменовать указатель после увеличения адреса, который он содержит, скобки необходимы, поскольку приоритет операции разыменования выше, чем приоритет арифметических операций + или -. Если вы напишете выражение *pdata* ***+*** *1* вместо \* *(pdata* ***+*** *1****)***, это добавит единицу к значению, находящемуся по адресу, хранящемуся в *pdata*, что эквивалентно выполнению

*data [2] + 1*.

Можно использовать имя массива, как если бы это был указатель, для обращения к его элементам. Если у вас есть одномерный массив вроде того, что раньше, объявленный, как:

*long data[5];*

то, применив нотацию указателя, вы можете сослаться на элемент *data [3]*, напри­мер, так: \* *(data* ***+*** *3)*. Этот вид нотации может применяться совершенно свободно, так что для доступа к элементам *data [ 0 ], data [ 1 ], data [ 2 ]* вы можете писать *\*data, \*(data + 1), \* (data+2)* и так далее.