1.Переменная символьного типа, её объявления и инициализация. Область использования переменного типа.

Переменные используются для хранения значений (sic!). Переменная характеризуется типом и именем. Начнём с имени. В си переменная может начинаться с подчерка или буквы, но не с числа. Переменная может включать в себя символы английского алфавита, цифры и знак подчёркивания. Переменная не должна совпадать с ключевыми словами (это специальные слова, которые используются в качестве управляющих конструкций, для определения типов и т.п.)

|  |  |  |  |
| --- | --- | --- | --- |
| auto | double | int | struct |
| break | else | long | switch |
| register | typedef | char | extern |
| return | void | case | float |
| unsigned | default | for | signed |
| union | do | if | sizeof |
| volatile | continue | enum | short |
| while | inline |  |  |

А также ряд других слов, специфичных для данной версии компилятора, например far, near, tiny, huge, asm, asm\_ и пр.

Например, правильные идентификаторы  
a, \_, \_1\_, Sarkasm, a\_long\_variable, aLongVariable, var19, defaultX, char\_type  
неверные  
1a, $value, a-long-value, short

Си - регистрозависимый язык. Переменные с именами a и A, или end и END, или perfectDark и PerfectDarK – это различные переменные.

Типы переменных

Тип переменной определяет

1) Размер переменной в байтах (сколько байт памяти выделит компьютер для хранения значения)

2) Представление переменной в памяти (как в двоичном виде будут расположены биты в выделенной области памяти).

В си несколько основных типов. Разделим их на две группы - целые и числа с плавающей точкой.

Целые

char - размер 1 байт. Всегда! Это нужно запомнить.

short - размер 2 байта

int - размер 4 байта

long - размер 4 байта

long long - размер 8 байт.

Здесь следует сделать замечание. Размер переменных в си не определён явно, как размер в байтах. В стандарте только указано, что  
  
char <= short <= int <= long <= long long  
  
Указанные выше значения характерны для компилятора VC2012 на 32-разрядной машине. Так что, если ваша программа зависит от размера переменной, не поленитесь узнать её размер.

Теперь давайте определим максимальное и минимальное число, которое может хранить переменная каждого из типов. Числа могут быть как положительными, так и отрицательными. Отрицательные числа используют один бит для хранения знака. Иногда знак необходим (например, храним счёт в банке, температуру, координату и т.д.), а иногда в нём нет необходимости (вес, размер массива, возраст человека и т.д.). Для этого в си используется модификатор типа signed и unsigned. unsigned char - все 8 бит под число, итого имеем набор чисел от 00000000 до 11111111 в двоичном виде, то есть от 0 до 255 signed char от -128 до 128. В си переменные по умолчанию со знаком. Поэтому запись char и signed char эквивалентны.

2.Строковая переменная типа Char. В чем ее отличие от переменной символьного типа?

Тип CHAR (символьный или строковый или литерный).  
  Его значениями являются отдельные символы: буквы, цифры, знаки. Символьные константы заключаются в кавычки, например, 'A', 'B', 'C', '4', '7', ' '(пробел).  
  
  Символьные переменные описываются предложением:  
  
  Var имя переменной: char;  
  
  Символьные значения можно вводить и выводить, присваивать, сравнивать. Ниже приведен пример, где выполняются все эти действия.  
  
Var  x,y:char;  
Begin  
  Write('Введите символ');     
  Readln(x);  
  Y:='A';  
  If x<y then write ('X') else write ('y');  
  {на экран буде выдан символ хранящийся в переменной  
  Х или Y в зависимости от проверки условия}  
  Readln;  
End.  
  
  Сравнивать символы можно благодаря тому, что в машинной памяти они хранятся в виде целых чисел (кодов символов). Из двух символов большим считается тот, код которого больше. Символы упорядочены следующим образом:  
  
'A'<'B'<...<'Z'  
'a'<'b'<...<'z'  
'0'<'1'<...<'9'  
'а'<'б'<...<'я'  
'А'<'Б'<...<'Я'  
  
Для символов допустимы все шесть операций сравнения: =, <=,>=,<,>,<>.

3.Способы объявления строковой переменной типа Char.

Строка — последовательность (массив) символов. Если в выражении встречается одиночный символ, он должен быть заключен в одинарные кавычки. При использовании в выражениях строка заключается в двойные кавычки. Признаком конца строки является нулевой символ \0. В C++ строки можно описать с помощью [массива](https://kvodo.ru/urok-7-1-massivyi-obshhie-svedeniya.html) символов (массив элементов типа char), в котором следует предусмотреть место для хранения признака конца строки.

4.Как осуществляется доступ к отдельным символам строковой переменной.

5.Сколько байт содержится в строковой переменной и сколько значащих символов?

Текстовые строки представляются массивами символов. Строковая переменная содержит на самом деле адрес этого массива. В отличие от символа, который занимает либо один, либо два байта в зависимости от используемой кодировки, строка имеет переменную длину. Существуют два способа указания длины строки:

строка заканчивается символом с нулевым кодом, т.е. либо нулевым байтом в случае однобайтового представления символов, либо двумя нулевыми байтами в случае двухбайтового представления. Такой способ принят в языке Си. Отметим, что нулевой байт - это вовсе не символ '0'! Символ '0' имеет код 48 в кодировках ASCII и UNICODE, а изображаемых символов с нулевым кодом не существует;

строка в качестве первого элемента (байта или двух байтов) содержит общее число символов, не включая начального элемента. Затем идут сами символы в указанном количестве. Такой способ используется в языке Паскаль.

Недостаток первого способа состоит в том, что для вычисления длины строки необходимо последовательно просмотреть все ее элементы, начиная с первого, пока не будет найден нулевой байт. Такая операция может быть долгой для длинной строки. Недостаток второго способа заключается в том, что длина строки ограничена. В случае однобайтовых символов максимальная длина строки равна 255, т.е. максимальному числу, которое можно записать в одном байте. Длина строки двухбайтовых символов ограничена числом 65535.

Впрочем, существуют и другие способы представления строк, которые используются в объектно-ориентированных языках. Строка рассматривается как объект, внутреннее устройство которого скрыто от пользователя, хотя, как правило, он содержит массив или адрес массива символов и длину строки. Обычно в случае представления строк в виде объектов ограничения на длину строки отсутствуют.

6.Каким элементом заканчивается строковая переменная типа Char?

Окончательный элемент - 0 будет только в массиве символов, массивы остальных типов данных оканчиваются последним элементом массива.

7. Какая функция используется для склеивания двух строк типа Char?

Для соединения строк в одну строку используются функции strcat и strncat.

8. Поясните особенности склеивания строковых переменных типа Char?

Функция

                char\*  strcat ( char  \*str1, const  char  \*str2 );

присоединяет строку str2 к строке str1, причем завершающий нулевой байт строки str1 стирается. Функция возвращает указатель на строку str1.

9. Как скопировать в строку типа Char какой-либо текст?

Строки могут копироваться одна в другую. Для этого можно использовать функцию strcpy, которая имеет прототип:

char\* strcpy(char\* str1, const char\* str2);

Функция выполняет побайтное копирование строки, на которую указывает str1. Копирование прекращается только в случае достижения символа конца строки (\0) str2, поэтому перед копированием необходимо убедиться в том, что длина строки str2 не больше длины строки str1. В противном случае возможны ошибки в программе.

Рассмотрим пример. Следующий фрагмент программы копирует в строку z слово "September".

char z[10];

strcpy(z,"September");

Можно проводить копирование не всей строки, а отдельного ее фрагмента до конца строки.

10. Какая функция преобразует символы строки типа Char к верхнему регистру. Функция char\* isPstrAlfabit (char\* str, int& n) получает строку str и ищет в ней первую подстроку, содержащую только символы латиницы (строчные или прописные буквы латинского алфавита). Если такая подстрока найдена, функция возвращает через параметр по ссылке длину подстроки и ее адрес через возвращаемое значение, и 0 в противном случае. Используйте при разработке алгоритма функции функцию isLatin.

Функция int UpDown (char& s) получает через параметр по ссылке символьный объект, преобразует в нем, если это прописной буквенный символ в символ строчный и возвращает через возвращаемое значение 1, если преобразование сделано, и 0 в противном случае.

Функция char\* UpDownNPstr (char\* str, int n) получает строку str и преобразует каждую n-ую подстроку латинских букв в подстроку тех же букв, но в коде верхнего регистра. Функция возвращает адрес строки str, если преобразования были осуществлены хотя бы над одной n-ой подстрокой строки str, и 0, если ни один символ в строке не был подвергнут преобразованию. Функция использует разработанные Вами функции isPstrAlfabit и UpDown.

Программа со стандартного устройства ввода считывает число n и строку, и, используя функцию UpDownNPstr, преобразует строку, выдавая на экран только те из них, в которых удалось сделать преобразования. Программа подсчитывает количество строк, подвергшихся изменению.

11. Как определить длину строки типа Char.

Для определения длины строки используется функция strlen.

                Функция

                size\_t  strlen ( const  char  \*str);

возвращает длину строки, не учитывая последний нулевой байт. Например,

                char  str[] = "123";

                printf ("len = %d\n", strlen ( str ));                   // печатает: len = 3

12. Определите в чем особенность строковых переменных типа AnsiString?

AnsiString - класс динамической строки емкостью до 232-1 символов. Базой для создания этого класса послужил паскалевский тип String, который был расширен в соответствии с возможностями C++.

Особенностью его является то, что два экземпляра этого класса могут физически занимать один и тот же участок памяти. Экземпляр этого класса содержит счетчик ссылок к нему, когда этот счетчик обнуляется, экземпляр автоматически уничтожается. Если попытаться изменить экземпляр, к которому имеется более одной ссылки, то будет создана новая копия этой строки, которая и будет изменяться. Это уменьшает портебность программы в памяти, но и уменьшает скорость обработки.

Наследием Паскаля является также то, что символы в нем нумеруются с 1, а не с 0, как это принято в C/C++

13. Какие операции определены для строк типа AnsiString?

Для AnsiString определены операции отношения ==, !=, >, <, >=, <=. Сравнение производится с учетом регистра.

14. Как объявляется и инициализируется строка типа AnsiString?

Создание и объявление строк

       Операции со строками в C++ Builder главным образом используют класс AnsiString. Класс AnsiString не является производным от TObject. Поэтому, у него высокая степень независимости и гибкости от приложения или элементов управления.

       Уровень поддержки работы, обеспечиваемый библиотекой VCL, строковых операций и связанных с ними текстовых операций, чрезвычайно высок. Поддерживается почти любая операция со строками. Так как существуют множество функций, то мы будем рассматривать те, что наиболее часто используются. Тем не менее, все библиотечные функции имеют большое значение и могут избавить вас от написания лишнего кода и прочих проблем.

       Многие элементы управления используют свойства AnsiString. Все элементы управления, имеющие заголовок (формы, панели, метки и др.), обладают свойством Caption, с установленным значение типа AnsiString. Многие другие элементы управления, такие как  edit box используют класс AnsiString как основу для работы с текстом. Основываясь на этих двух фактах, вы можете использовать и устанавливать значения типа AnsiString. В некоторых других случаях, возможно, придется объявить и инициализировать строку перед её использованием.

       Для объявлении строки поставьте слово AnsiString перед именем. Например:

       AnsiString Country;

       У класса есть собственный конструктор и можно объявить его переменной с пустыми скобками, которые могут вызвать конструктор класса. Например:

       AnsiString City();

Иницализация строки

       Существуют два варианта инициализации переменной типа AnsiString. После её объявления, можно назначить нужное значение с помощью оператора присваивания. Например:

              AnsiString Country;

       Country = "Madagascar"

        ;

       Также можно инициализировать строковую переменную при её объявлении, снова используя оператор присваивания и определяя нужное значение. Например:

       AnsiString Province("British Columbia");

       После того как вы определили строку, вы можете использовать её по своему усмотрению. Например, можно изменить заголовок элемента управления:

       void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

       {

              AnsiString Country;

              Country = "Madagascar";

              Panel1->Caption = Country;

       }

Вы также можете использовать строку для занесения её в сам элемент управления:

       void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

       {

              AnsiString City = "Antananrivo";

              Edit1->Text = City;

       }

15. Как преобразовать строку типа AnsiString в строку типа Char?

реобразование типов

Как правило, самый частый вопрос, который возникает при работе со строками, — это преобразование типов, когда из AnsiString нужно получить массив Char или наоборот.

Итак, сначала получаем из AnsiString массив Char. Сделать это можно следующим образом:

Вариант 1

|  |
| --- |
| AnsiString AStr="Hello";  char Str1[10]={'a','a','a','a','a','a','a','a','a','a'};    strcpy(Str1,AStr.c\_str()); // копируем в существующий массив    Edit1->Text=Str1; |

Вариант 2

|  |
| --- |
| AnsiString AStr="Hello";  char \*Str2; // создаём указатель, который пока ни к чему не привязан    Str2 = new char[ Astr.Length() + 1 ]; // создаём новый массив,  // привязываем к нему указатель  strcpy(Str2,AStr.c\_str()); // и копируем в этот массив    Edit1->Text=Str2; |

В первом случае нужно дополнительно проверить, чтобы массив, в который происходит копирование, был достаточного размера. Размер массива должен быть на 1 больше количества символов в строке (1 дополнительный элемент нужен для нультерминатора).

Как делать не нужно!

Вариант 2

|  |
| --- |
| AnsiString AStr="Hello";  char \*Str2 = AStr.c\_str(); |

В хэлповнике по Builder-у написано, что функция c\_str() возвращает временный указатель на внутренний буфер строки в объекте AnsiString, который действителен только на время выполнения выражения, в котором он используется.

Для того, чтобы преобразовать массив char-ов в AnsiString можно использовать вот такой метод:

|  |
| --- |
| char Str1[]={'H','e','l','l','o','  char Str1[]={'H','e','l','l','o','\0'};  AnsiString AStr;    AStr=AnsiString(Str1);  '};  AnsiString AStr;    AStr=AnsiString(Str1); |

Преобразование выполняется до первого встреченного нультерминатора. То есть, если массив выглядел, скажем, вот так: {‘H’,’e’,’\0′,’l’,’o’,’\0′}, то после преобразования его в AnsiString, полученная строка будет выглядеть так: «He».

16. В каких случаях необходимо использовать строки типа AnsiString и типа Char?

Итак, в С++ Builder строки можно задать тремя разными способами: через специальный класс AnsiString, через массив символов char[] и через указатель на первый символ массива char\*:

|  |
| --- |
| AnsiString String1="Hello!";  char \*String2="Hello!";  char String3[]={'H','e','l','l','o','!','  AnsiString String1="Hello!";  char \*String2="Hello!";  char String3[]={'H','e','l','l','o','!','\0'};  char String4[7]={'H','e','l','l','o','!','\0'};  '};  char String4[7]={'H','e','l','l','o','!','  AnsiString String1="Hello!";  char \*String2="Hello!";  char String3[]={'H','e','l','l','o','!','\0'};  char String4[7]={'H','e','l','l','o','!','\0'};  '}; |

Задание строк String3 и String4 на самом деле эквивалентно, просто если задавать строку так, как у нас задана строка String3, то её обязательно нужно сразу инициализировать, тогда циферку в квадратные скобки за вас вставит компилятор.

Использование класса AnsiString удобно тем, что заданные с его помощью строки являются динамическими и можно особо не париться с их размером. То есть можно писать, например, так:

|  |
| --- |
| AnsiString String1;  String1="Hello!";  String1="Hello world!"; |

И пофигу, что во втором случае строка стала длиннее, пускай Builder сам с этим разбирается.

Для символьных массивов такой фокус не прокатит от слова совсем. Для указателя прокатит, но нужно понимать, что при этом на самом деле не меняется исходный массив, а просто изменяется значение указателя и он начинает указывать на другой существующий массив (который кто-то создал, память под него выделил). При этом старый массив, на который указывал указатель, просто потеряется. Это чревато всякими утечками памяти и тому подобными косяками.

Строки, заданные как массив символов, должны оканчиваться нулём (по-другому их называют нультерминированные). Массив Char-ов может содержать нули, но при этом строкой считается всё, что содержится от начала массива до первого встреченного нулевого символа. Например, можно задать вот такой массив:

|  |
| --- |
| char TempStr[]={'H','e','l','l','o',0x0,'w','o','r','l','d','  char TempStr[]={'H','e','l','l','o',0x0,'w','o','r','l','d','\0'};  '}; |

Если в этом случае попробовать посчитать длину строки TempStr, то она окажется равной пяти, то есть количеству элементов от начала строки до нультерминатора, без учёта его самого. (Как видите, элементы массива не обязательно писать в кавычках, как символы, можно и напрямую в hex).

Более того, один массив может содержать сразу две строки. Фактически в предыдущем примере так оно и есть. Если мы напишем вот такой код:

|  |
| --- |
| char TempStr[]={'H','e','l','l','o',0x0,'w','o','r','l','d','  char TempStr[]={'H','e','l','l','o',0x0,'w','o','r','l','d','\0'};  Edit1->Text=TempStr;  Edit2->Text=&TempStr[6]; // &TempStr[6] - адрес шестого элемента ('w')  '};  Edit1->Text=TempStr;  Edit2->Text=&TempStr[6]; // &TempStr[6] - адрес шестого элемента ('w') |

, то в Edit1 получим текст «Hello», а в Edit2 — текст «world».

Ещё одной фишкой является то, что в AnsiString элементы считаются с единицы, а в массивах символов (да и в любых вообще массивах) — с нуля (именно поэтому символ ‘w’ в предыдущем примере шестой, а не седьмой). То есть вот такой код:

|  |
| --- |
| char Str[]={'H','e','l','l','o','  char Str[]={'H','e','l','l','o','\0'};  AnsiString AStr="Hello";  Edit1->Text=Str[1];  Edit2->Text=AStr[1];  '};  AnsiString AStr="Hello";  Edit1->Text=Str[1];  Edit2->Text=AStr[1]; |

выведет в поле Edit1 символ ‘e’, а в поле Edit2 символ ‘H’.

Да, кстати, как видите, элементы AnsiString тоже можно получить по индексам, совсем как элементы массива.

Поскольку AnsiString — это класс, то он имеет множество всяких полезных свойств и методов, позволяющих особо не париться с различными операциями над строками. Для массива вместо этого приходится пользоваться специальными библиотечными функциями или писать свои.

18. Как склеить несколько строк типа AnsiString?

Следующий оператор демонстрирует конкатенацию (склеивание) двух строк:

Label1->Caption = Edit1->Text + ' ' + Edit2->Text;

В данном случае в свойстве Label1->Caption отображается текст, введенный пользователем в окне редактирования Edit1, затем записывается символ пробела, а за тем текст, введенный в окне редактирования Edit2. Склеивание строк типа AnsiString легко осуществляется перегруженной операцией сложения "+".

19. Как можно осуществить доступ к отдельным символам строки типа AnsiString?

Получить символ из строки типа AnsiString можно с помощью индексного оператора:

AnsiString ansi\_string = "Строка";

char character = ansi\_string[1]; /\* получаем самый первый символ \*/

ansi\_string[6] = 'и'; /\* заменим 6-й на букву 'и' \*/

20. Что такое структуры в языке C++ Builder?

Из предыдущих глав известно, что массив это переменная для хранения множества данных одного типа. Если возникает необходимость обрабатывать разнородную информацию как единое целое, то применяют тип данных структуры. Он позволяет сгруппировать объекты различных типов данных под одним именем.

Для того, чтобы объявить переменные структурного типа, вначале нужно задать новый тип данных, указав имя структуры и её элементы. Элементы структуры называются полями, и могут иметь любой тип данных кроме типа этой же структуры. Далее приведён пример создания структурного типа student, полями которого являются фамилия студента, шифр группы, год начала обучения и оценки по четырём предметам:

struct student

{

//Поля структуры:

char fio [ 30 ];

char group [ 8 ];

int year;

int informatika, math, fizika, history;

}

На основании созданного структурного типа данных можно описать переменные типа student:

student Vasya; //Переменная Vasya типа student.

student ES [ 50 ]; //Массив, элементы которого имеют тип student.

student \*x; //Указатель на тип данных student

Обращаются к полям переменной структурного типа так:

имя\_структуры.поле

Например,

Vasya.year; //Обращение к полю year переменной Vasya.

ES[4].math; //Обращение к полю math элемента ES[4].

21. Как объявляется, определяются и используются структуры?

Поскольку структуры определяются программистом, то вначале мы должны сообщить компилятору, как она вообще будет выглядеть. Для этого используется ключевое

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | struct Employee  {      short id;      int age;      double salary;  }; |

Мы определили структуру с именем Employee. Она содержит 3 переменные: id типа short, age типа int и salary типа double. Эти переменные, которые являются частью структуры, называются членами структуры (или ещё «полями структуры»). Employee — это простое объявление структуры. Хоть мы и указали компилятору, что она имеет переменные-члены, память под неё сейчас не выделяется. Имена структур принято писать с заглавной буквы, чтобы отличать их от имён переменных.

Предупреждение: Одной из самых простых ошибок в C++ является забыть точку с запятой в конце объявления структуры. Это приведёт к ошибке компиляции в следующей строке кода. Современные компиляторы, такие как Visual Studio 2010 и выше, укажут вам, что вы забыли точку с запятой в конце, но более старые компиляторы могут этого и не сделать, из-за чего такую ошибку будет трудно найти. О том, как установить Visual Studio или какую выбрать IDE мы уже говорили в [уроке №4](https://ravesli.com/urok-4-ustanovka-integrirovannoj-sredy-razrabotki-ide-s/).

Чтобы использовать структуру Employee, нам нужно просто объявить переменную типа Employee:

|  |  |
| --- | --- |
| 1 | Employee john; // имя структуры Employee начинается с заглавной буквы, а переменная john с маленькой |

Здесь мы определили переменную типа Employee с именем john. Как и в случае с обычными переменными, определение переменной структуры приведёт к выделению для неё памяти.

Объявить можно и несколько переменных одной структуры:

|  |  |
| --- | --- |
| 1  2 | Employee john; // создаём отдельную структуру Employee для John-a  Employee james; // создаём отдельную структуру Employee для James-a |

22. Чем структуры отличаются от классов.

В С++ классом называются типы объявленные с помощью class, struct или union.

Это означает что в С++ есть только классы, и все правила для class и struct одинаковы, за исключением случаев где отличия указаны явно.

Отличий class от struct всего два

Member access control [class.access]

Члены класса, определенного с помощью ключевого слова class, по умолчанию являются private. Члены класса, определенного с помощью ключевого слова struct или union, по умолчанию являются public.

Accessibility of base classes and base class members [class.access.base]

При отсутствии спецификатора доступа (т.е. private/protected/public) у базового класса, базовый класс будет public если класс определен с помощью struct и private если класс определен с помощью class.

(Хотя union и является классом, он имеет много других отличий от class и struct, что выходит за рамки этого вопроса.)

Использование

Есть популярное мнение, что для "расово правильных ООП классов" надо использовать только class, а для структур данных - только struct, и еще у struct не должно быть методов. Это не более чем вкусовщина, и подобные вещи должны быть закреплены в руководствах по стилю кодирования, как например это сделано в [Google C++ Style Guide](http://google-styleguide.googlecode.com/svn/trunk/cppguide.html" \l "Structs_vs._Classes).

В коротких примерах кода, предпочтительнее использовать struct, и не использовать private/public, если они не относятся к сути проблемы.

23. Какие свойства объекта StringGrid определяется число столбцов и строк?

Компонент [StringGrid](https://cubook.pro/object/tablitsy-izobrazhenij-drawgrid-i-stringgrid" \o "Таблицы изображений) представляет собой таблицу, содержащую строки и столбцы. Данные таблицы могут быть только для чтения или редактирования. Таблица может иметь полосы прокрутки, причем заданное число первых строк и столбцов может быть фиксированным и не прокручиваться. Таким образом, можно задать заголовки столбцов и строк, постоянно присутствующие в окне компонента. Каждой ячейке таблицы может быть поставлен в соответствие некоторый объект.

Компонент StringGrid предназначен в первую очередь для отображения таблиц текстовой информации. Однако компонент может отображать и графическую информацию.

Все эти свойства доступны во время выполнения. Задавать тексты можно программно или по отдельным ячейкам, или сразу по столбцам и строкам с помощью методов класса TStrings.

Свойства ColCount и RowCount определяют соответственно число столбцов и строк, свойства FixedCols и FixedRows - число фиксированных, не прокручиваемых столбцов и строк. Цвет фона фиксированных ячеек определяется свойством FixedColor.

Свойства LeftCol и TopRow определяют соответственно индексы первого видимого на экране в данный момент прокручиваемого столбца и первой видимой прокручиваемой строки.

Свойство ScrollBars определяет наличие в таблице полос прокрутки. Причем полосы прокрутки появляются и исчезают автоматически в зависимости от того, помещается таблица в соответствующий размер или нет.

Свойство Options является множеством, определяющим многие свойства таблицы:

наличие разделительных вертикальных и горизонтальных линий в фиксированных (goFixedVertLine и goFixedHorzLine) и не фиксированных (goVertLine и goHorzLine) ячейках;

возможность для пользователя изменять с помощью мыши размеры столбцов и строк (goCoISizing и goRowSizing);

перемещать столбцы и строки (goColMoving и goRowMoving) и многое другое.

Важным элементом в свойстве Options является goEditing - возможность редактировать содержимое таблицы.

В основном компонент StringGrid используется для выбора пользователем каких-то значений, отображенных в ячейках.  
Свойства Col и Row показывают индексы столбца и строки выделенной ячейки. Возможно также выделение пользователем множества ячеек, строк и столбцов.  
Среди множества событий компонента StringGrid следует отметить событие OnSelectCell, возникающее в момент выбора пользователем ячейки. В обработчик этого события передаются целые параметры ACol и ARow - столбец и строка выделенной ячейки, и булев параметр CanSelect - допустимость выбора.

Параметр CanSelect можно использовать для запрета выделения ячейки, задав его значение false. А параметры ACol и ARow могут использоваться для какой-то реакции программы на выделение пользователя.

Пример №1

Оператор выведет в метку Label1 сообщение о строке, столбце и тексте выбранной ячейки:

Label1->Caption = "Выбрана ячейка " + IntToStr(ARow) + ":" + IntToStr(ACol) + ". Текст: " + StringGrid1->Cells[ACol][ARow];

Пример №2

Данный пример обеспечит заполнение ячеек таблицы текстами:

int i, j;

for(i = 1; i < StringGrid1->ColCount; i++)

StringGrid1->Cells[i][0] = "столбец " + IntToStr(i);

for(i = 1; i < StringGrid1->RowCount; i++)

{

StringGrid1->Cells[0][i] = "строка " + IntToStr(i);

for(j = 1; j < StringGrid1->RowCount; j++)

{

StringGrid1->Cells[j][i] = IntToStr(i) + "+" + IntToStr(j);

}

}

24. Какие свойство используется для открытия доступа к редактированию таблицы?

Данный обработчик прячет кнопку, вызвавшую событие ее нажатия:

void \_\_fastcall

TForm1::Button1Click(TObject \*Sender)

{

TButton \*ptButton;

ptButton=(TButton\*)Sender;

ptButton->Visible=false;

}

Этот обработчик можно назначить сразу нескольким компонентам, но только TButton. При попытке преобразования к другого типу компонента программа выдаст исключение на преобразование типов. Если же необходимо выполнить преобразование из неизвестного класса, то можно использовать конструкции try...catch, но проще воспользоваться динамическим преобразованием - оператором dynamic\_cast.