**Трансляция кода Delphi в код C++Builder**

**Введение**

Цель этой статьи состоит в том, чтобы помочь вам понять основные различия и подобия между C++ и Object Pascal (язык, используемый в Delphi от Borland), и помочь вам в преобразовании проекта написанного на Delphi в проект написанный на C++ (С++Builder от Borland).В жалбнейшем в этой статье вместо терминов Object Pascal и Delphi будет употребляться аббревиатура "OP".

Типы данных

|  |  |  |
| --- | --- | --- |
| Op | Размер или значение | С++ |
| Shortlnt | целое 8 бит | signed char |
| Smalllnt | целое 16 бит | short |
| Longlnt | целое 32 бита | int |
| Byte | целое без знака 8 бит | unsigned char |
| Word | целое без знака 16 бит | unsigned short |
| Integer | целое 32 бита | int |
| Cardinal | целое без знака 32 бита | unsigned int |
| Boolean | true,false или | bool |
| ByteBool | true,false или целое без знака 8 бит | unsigned char |
| WordBool | true,false или целое без знака 16 бит | unsigned short |
| LongBool | true,false или целое без знака 32 бита | BOOL (WinAPI) |
| AnsiChar | символ без знака 8 бит | char |
| WideChar | символ UNICODE размером в слово | wchar\_t |
| Char | символ без знака 8 бит | char |
| AnsiString | AnsiString Delphi | AnsiString |
| String[n] | прежний стиль строк Delphi, n= 1..255 бит | SmallString<n> |
| ShortString | прежний стиль строк Delphi, 255 бит | SmallString<255> |
| String | AnsiString Delphi | AnsiString |
| Single | число с плавающей запятой 32 бита | float |
| Double | число с плавающей запятой 64 бита | double |
| Extended | число с плавающей запятой 80 бит | long double |
| Real | число с плавающей запятой 32 бита | double |
| Pointer | родовой указатель 32 бита | void \* |
| PChar | указатель на символы 32 бита | unsigned char \* |
| PAnsiChar | указатель на символы ANSI 32 бита | unsigned char \* |
| Comp | число с плавающей запятой 64 бита | Comp |
| OleVariant | значение variant OLE | OleVariant |

**Ключевые слова**

Следует отметить, что язык С++ является регистро зависимым, в отличие от OP , который таковым не является.

|  |
| --- |
| C++ |
| #define #elif #else #endif #error #if #ifdef #ifndef #include #line #pragma #printf #undef auto break case catch char class const const\_cast continue default delete do double dynamic\_cast else enum except extern false finally float for friend goto if inline int long mutable namespace new operator private protected public register reinterpret\_cast return short signed sizeof static static\_cast struct switch template this throw true try typedef typeid union unsigned using virtual void volatile while xalloc new delete |
| OP |
| ABSOLUTE AND ARRAY ASM ASSEMBLER BOOLEAN BYTE CASE CHAR COMP CONST CONSTRUCTOR DESTRUCTOR DIV DO DOUBLE DOWNTO ELSE EXTENDED EXTERNAL FALSE FAR FILE FOR FORWARD FUNCTION GOTO IF IMPLEMENTATION IN INLINE INTEGER INTERFACE INTERRUPT LABEL LONGINT MOD NEAR NIL NOT OBJECT OF OPERATOR OR PACKED PRIVATE PROCEDURE PROGRAM REAL RECORD REPEAT SET SHL SHORTINT SHR SINGLE STRING TEXT THEN TO TRUE TYPE UNIT UNTIL USES VAR VIRTUAL WHILE WITH WORD XOR |

**Операторные признаки конца**

C++

Многие выражения заканчиваются точкой с запятой ;

Хотя имеется парочка исключений. Например при подключении заголовочных файлов точка с запятой не требуется

#include "foo.h" // не заканчивается точкой с запятой;

#define MAXNAMELENGTH 35 // также не заканчивается точкой с запятой;

OP

Все выражения заканчиваются точкой с запятой;

Объявление переменных

C++

Максимальная длина переменной 32 символа - может быть и больше но будут распознаны только первые 32 символа.

Переменные могут быть объявлены в любом месте кода, ну и соответственно переменные должны быть обявлены до того места где они будут использоваться.

// ... какойто код ...

char sName[10];

int iCount, iLoop, iValToReach;

double dMaxLoan, dTotal;

float fMaxRate = 123.875;

OP

Максимальная длина переменной 63 символа -может быть и больше но будут распознаны только первые 63 символа.

Переменные должны объявляться в блоке "var" вначале процедуры или функции или в определении объекта до объявления функций и процедур.Значения переменных не могут быть назначены внутри блока var .

// ... какойто код ...

function PayBack\_d(var dAmount: double): double ;

var

iCount, iLoop, iValToReach: integer ;

dMaxLoan, dTotal, dMaxRate: double ;

begin

dMaxRate := 123.875;

...

**Строки**

C++

В C++ отсутствует специальный тип строк. Строки рассматриваются как массивы символов, оканчивающиеся нулевым символом (''). Размер строки может быть неограничен(теоретически максимальный объем памяти) Строка доступна через указатель на первый символ в строке. Значением строки является адрес ее первого символа. Таким образом можно сказать, что в C++ строка является указателем — указателем на первый символ строки. В этом смысле строки подобны массивам, потому что массив тоже является указателем на свой первый элемент.Строка может быть объявлена либо как массив символов, либо как переменная типа char\*. Каждое из приведенных ниже объявлений верны:

char sName[26]; // 25 символов плюс NULL

char psDemo[] = "Hello, there!";

char \* psDemo = new char[26];

Для обработки строк имеется ряд библиотечных функций. Основные из них strcat — конкатенация (склеивание) двух строк, strcmp — сравнение двух строк, strcpy — копирование одной строки в другую, strstr — поиск в строке заданной подстроки, strlen — определение длины строки, strupr — преобразование символов строки к верхнему регистру, sprintf — построение строки по заданной строке форматирования и списку аргументов и ряд других функций.

OP

Delphi поддерживает строки практически неограниченной длины (до 2 Гб), в дополнение к 255-символьным строкам, известным как pascal-строки. Новая директива компилятора $Н позволяет контролировать смысл использования зарезервированного слова string. Состояние этой директивы по умолчанию ($Н+) подразумевает использование в качестве типа string длинных строк. $Н- подразумевает использование коротких 255-символьных pascal-строк. Все компоненты Delphi используют длинные строки.Следует отметить, что, вне зависимости от состояния директивы $Н, если специфирована максимальная длина строки, меньшая 255, то подразумевается короткая строка (ShortString). Назначение длины строки с использованием квадратных скобок всегда подразумевает подтип ShortString. Длинные строки, введенные в Delphi 2.0, в большинстве случаев совместимы со строками старого типа. При этом необходимо принимать во внимание следующие особенности их использования:

Память для длинных строк выделяется динамически. Длинные строки являются указателями на строки, хранимые в хипе (heap). Когда вы изменяете длину строки, Delphi32 перераспределяет память. Поскольку менеджер памяти пытается перевыделить память в той же самой области, он не всегда может это сделать. Поэтому, для увеличения длины строки, следует использовать введенную в Delphi 2.0 процедуру SetLength, перераспределяющую память, если это возможно

Длинные строки автоматически заканчиваются #0, поэтому вы можете использовать те методы работы с этими строками, которые применяли для обработки null-terminated string. Поэтому теперь отсутствует необходимость использовать функции конвертации, такие как StrPCopy, хотя Delphi и поддерживает короткие строки

Delphi поддерживает счетчик использований для каждой длинной строки. Это значит, что если ваше приложение копирует строку (например, передавая ее в качестве строкового параметра в процедуру), то автоматически увеличивается количество ссылок на строку. Когда заканчивается использование очередной копии строки, то есть, происходит выход за область видимости, счетчик использований автоматически уменьшается. Когда счетчик использований становится равным нулю, выделенная под строку память освобождается автоматически

Поскольку реально длинные строки не дублируются, а для них поддерживаются счетчики использований, то присваивание длинных строк происходит намного быстрее по сравнению с короткими строками.Несмотря на то, что большая часть кода, используемого при работе со строками интероперабельно для коротких и длинных строк (то есть применимо к обоим видам строк), тем не менее, некоторые операции над короткими строками не применимы к длинным строкам. Длинные строки не могут передаваться в параметры типа OpenString или как var-параметры, соответствующие коротким строкам.

**Массивы**

Массивы - упорядоченные последовательности одного типа данных (могут включать также и объекты). Методы для объявления массива в C ++ и OP - различные.

С++

Массивы "основаны на нуле" - первый элемент - элемент [0], второй элемент элемент [1], третий - [2], и т.д. Это всегда запутывает новичков.

Объявление массива:

double Array[31];

Использование :

for(int i = 30, j = 29; i > 0; i--,j--)

Array[i] = Array[j];

OP

Начало массива в элементе 1... но не всегда.В Delphi встречаются компоненты и объекты "основаные на нуле" наподобие С++. Например TList элементы которого начинаются с нуля. Любой массив который вы объявите будет основан на единице

var ArrayS: array[1..31] of double;

i,j: integer;

begin

j := 30;

for i := 31 downto 2 do

begin

ArrayS[i] = ArrayS[j];

Dec(j); { или j := j-1; }

end;

......

Оба языка поддерживают многомерные массивы

С++

double dMatrix[50] [20];

OP

dMatrix: array[1..50, 1..20] of double;

Приравнивание и сравнение переменных

C++

Сравнение производиттся оператором "двойное равно" (==), приравнивание одним равно (=)

int i,j;

if(i==j)//сравнение

.....

i=j;//приравнивание

....

OP

Сравнение производиттся оператором "равно" (=), приравнивание двоеточие+одно равно (:=)

...

if dMaxRate = 15.75 then

begin ...

dMaxRate := 12.5;

...

end;

исключением из правил в OP является приравнивание переменной к константе. В этом случае применяется одиночное равно (=)

**Объявление констант**

Константы - значения, которые не изменяются . Идентификатор может быть объявлен как константа, и в C ++ и OP. И C ++ и OP константам должны быть назначены значения при их объявлении.

С++

При объявлении константы перед типом данных используется ключевое слово const

const int iMax = 2000;

const double dMaxValue = 1234.75;

В С++ константы также можно определять через директиву препроцессора #define

#define MAXA 1000

но на практике это устарело. Так как компилятор не может выполнить контроля соответствия типов(он не может определить что это ini или double)

OP

Константы, подобно переменным, должны быть объявлены в блоке кода "const" в начале определения процедуры или функции ,определения или в декларации объекта .

function PayBack\_d(var dAmount: double): double;

const iMAX = 2000; dMAXVALUE = 1234.75;

var iCount, iLoop, iValToReach: integer;

dMaxLoan, dTotal, dMaxRate: double;

begin

dMaxRate := dMAXVALUE;

{...}

Функции и процедуры

Блоки Кода, которые исполняют определенную задачу в C ++, называются "функциями" независимо от того, действительно ли они возвращают значение. В OP, функции должны возвращать значение, процедуры не возвращают значение. В C ++, все функции должны иметь прототип, объявленный перед применением (так что транслятор может сравнивать обе функции , чтобы гарантировать соответствие). В OP, однако, функция или определение процедуры могут опустить (но это не желательно) параметры и опускать тип возвращения.Декларация функции в C ++ и определений требует фигурных скобок () независимо от того, действительно ли передаются любые параметры. OP функции и процедуры не нуждаются в фигурных скобках () при их объявлении или определении , если не передается никаких параметров,для вызова функции или процедуры.

С++

Декларации

void Foo(double X);

int Foo2();

Реализация

void Foo(double X)

{

int Y=X;

}

int Foo2()

{

return 0;

}

OP

Каждая функция и процедура должны быть идентифицированы включением ключевого слова "function" или "procedure" в начале каждого определения .

function IntSinceLastAddDate\_d(var dAvailCash: double): double;

procedure SetNewIntRate; {нет параметров и возвращаемого значения}

...

function LoanC.IntSinceLastAddDate\_d(var dAvailCash: double): double;

var dSomething: double;

begin

...

result := dSomething; {глобальная переменная "result" назанчается возвращаемым значением!}

end;

procedure LoanC.SetNewIntRate;

begin

...

end;

И C ++ и OP могут передавать параметры по значению или по ссылке или передавать константы...

C++ по значению ... double IntSinceLastAddDate\_d(double dAvailCash);

OP по значению ... function IntSinceLastAddDate\_d(dAvailCash: double): double;

C++ по ссылке ... double IntSinceLastAddDate\_d(double &dAvailCash);

OP по ссылке ... function IntSinceLastAddDate\_d(var dAvailCash: double): double;

C++ константы... double IntSinceLastAddDate\_d(const double dAvailCash);

OP константы... function IntSinceLastAddDate\_d(const dAvailCash: double): double;

Конструкция with ... do

Вообще, C ++ более компактен чем OP. Но, C ++ не имеет инструкцию with ... do с... Это неудачно, потому что это действительно - большая особенность OP. C ++ код будет более подробным по сравнению с OP.

В C ++, когда Вы должны обратиться к членам данных, вы заканчиваете оператор подобно этому...

poC.oStock.aoTradesLast130Days[0].lVol = 0;

poC.oStock.aoTradesLast130Days[0].dHigh = 0;

poC.oStock.aoTradesLast130Days[0].dLow = 0;

poC.oStock.aoTradesLast130Days[0].dClose = 0;

Но в OP, Вы можете делать это гораздо менее читаемым, делая так...

with poC.oStock.aoTradesLast130Days[0] do begin

lVol := 0;

dHigh := 0;

dLow := 0;

dClose := 0;

end;

В принципе в данный момент это выглядит довольно коротко и ясно по стравнению с С++. Но в случае многочисленных вложеных операторов порой возникает путаница что с чем взаимодействует. Хотя... как говорят программисты, которые пишут на OP, им гораздо легче ориентироваться именно в таком типе записи (OP) чем в типе С++.

Комментарии

C++ поддерживает 2 вида комментариев:

Однострочный комментарий

//Это комментарий

И многострочный комментарий

/\*

А я многострочный комментарий, хотя и нахожусь всего на одной строке. Зато занял места на 3 строки.

\*/

OP в отличие от С++ поддерживает 3 вида комментариев. Это связано с тем, что Borland при написании Delphi внес в язык Pascal очень много элементов из C++.В часности добавился однострочный комментарий как в С++

Однострочный комментарий

//Это комментарий

И многострочный комментарий

(\*

А я многострочный комментарий, хотя и нахожусь всего на одной строке. Зато занял места на 3 строки.

\*)

{

И я многострочный комментарий, но работаю только в OP

}

Управляющие конструкции (if, else, switch...)

Существует пять структур управления на обоих языках, и они весьма подобны. Этот обзор займет много места .

1. Конструкция if ... else

C++

if(<логическое выражение>) // в скобках единственное выражение

{

...

}

else if(<логическое выражение>)

{

...

}

else

{

...

}

OP

if <логическое выражение> then

begin

{единственное выражение не должно находиться в скобках}

...

end

else if <логическое выражение> then

begin

....

end

else

begin

...

end; {только самый последний "end" в выражении терминируется точкой с запятой}

2. Конструкция switch/case

C++

switch(<целочисленная константа>)

{

case iX: ... break;

case iY: ... break;

default: ...

}

OP

case <целочисленная выражение переменная или константа> of

{здесь "begin" не нужен}

iX:

begin

...

end; {точка с запятой после каждого "end";}

iY:

begin

...

end;

else {двоеточие не требуется}

begin

...

end;

end;{вот здесь "end;" }

3. Цикл for

C++

for(iCount = 0; iCount <= 10; iCount++)

{

// инкреметн нужен для цикла, iCount++,может быть увеличен

// на любое значние не только на 1

...

break; // для выхода из цикла

continue; // для продолжения цикла

...

}

OP

for iCount := 1 to 10 do

begin

... {цикл for может увеличивать счетчик только на 1}

break; { для выхода из цикла }

continue; { для продолжения цикла }

...

end ;

Есть одно различие между выполнением "break" и "continue" в языках. В C ++, break, и continue - ключевые слова - часть языка;

В OP это библиотечные процедуры.Хотя используются также.

4. Цикл while

C++

while(<логическое выражение>)

{

// выражение проверяется в самом начале цикла,так что код может никогда не использоваться

// если выражение вернет false

...

break; // для выхода из цикла

continue; // для продолжения цикла

...

}

OP

while <логическое выражение> do

begin

{ выражение проверяется в самом начале цикла,так что код может никогда не использоваться

если выражение вернет false }

...

break; { для выхода из цикла }

continue; { для продолжения цикла }

...

end;

5. Цикл do ... while

5) The do/repeat ... loop construct:

C++

do

{

// выполняется некий код, затем проверяется выражение. В этом случае код выполнится хотябы один раз

...

break; // для выхода из цикла

continue; // для продолжения цикла

...

}while(<логическое выражение>);

OP

repeat

{ eвыполняется некий код, затем проверяется выражение. В этом случае код выполнится хотябы один раз

заметьте что в цикле нет ключевых слов begin ... end }

...

break; { для выхода из цикла }

continue; { для продолжения цикла }

...

until <логическое выражение>;