【B3】C++テクニカルセッション

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

C++Builder次期バージョン"2009"活用法

エンバカデロ・テクノロジーズエヴァンジェリスト
高橋 智宏

アジェンダ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++0x
- Delphi 2009の言語拡張とC++Builder 2009
- AnsiString, UTF8String, WideString, UnicodeString
- IDEへの追加機能
  - プリコンパイル済みヘッダーウィザード と 暗黙のインジェクト
  - 国際化支援機能 - ITE/ETM

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

追加されたC++0x機能

追加されたC++0x機能

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Strongly typed enumerations
- Explicit Conversion Operators
- static assertions
- Type trait functions
- alignof operator on types
- extern templates
- Variadic templates
- Rvalue references
- decltype
- New Unicode char types (char16_t and char32_t)
- Attributes - noreturn and final

- nullptr … ×（NULLの代わりですが、今回は実装されませんでした…）

強く型付けされたEnum

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- enum class 列挙型名 { …. };
  - 従来のenumはそのまま利用可能
  - 新しいenumではスコープの明示が必要
  - int型などとの相互変換には明示的なキャストが必要になる（予定）

```
enum class MyEnum { A = 0, B, C, D };

void enumtest()
{
  MyEnum e1 = A;          // ×
  MyEnum e2 = MyEnum::A;  // ○
  MyEnum e3 = 1;          // 未実装。今後のアップデートに期待
  int i = MyEnum::A;      // 未実装。今後のアップデートに期待
}
```


明示的なキャスト演算子

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- explicit operator XXX(){ … }
  - これまでは、explicitコンストラクタだけでした
  - explicit修飾子を追加
  - 明示的なキャストが必要

```
class MyCast {
public:
  explicit operator int(){ return 1; }
};

void casttest()
{
  MyCast c;
  int i = (int)c;
}
```

静的なアサーションによるチェック

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- static_assert (constant-expression, error-message);
  - constant-expression ー 静的に判定可能なbool値(true/false)
  - error-message ー コンパイルエラーのメッセージ
  - マクロではありません

```
template <typename T>
void asserttest(T x)
{
  static_assert(sizeof(T) == sizeof(long), "sizeof(T) must be sizeof(long)");
}
...
  double a1 = 11;
  asserttest(a1);
  long a2 = 22;
  asserttest(a2);
  int a3 = 33;
  asserttest(a3);
...
```


型特性を判定する関数

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- bool __is_XXXX(typename T, ...)
- bool __has_YYYY(typename T, ...)
  - コンパイル時および実行時に利用できる
  - 戻り値はbool(true/false)
  - 約50個の関数が利用可能 ー include¥dinkumware¥type_traits

```
#include <type_traits>
using namespace std;
template <typename T> void tt_test1(T x)
{
  static_assert(is_same<T, long>::value, "you must pass long!!");
  static_assert(__is_same(T, long), "you must pass long!!");

  bool r1 = is_same<T, long>::value;
  //bool r2 = __is_same<T, long>;
}
...
  long t1 = 0;
  tt_test1(t1);
  //int t2 = 0;
  //tt_test1(t2);
...
```

型特性を判定する関数(続き)

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- デストラクタにvirtualを要求させるには？

```
class MyParent {
public:
  //virtual ~MyParent() {}
  ~MyParent() {}
};
class MyChild : public MyParent {
public:
  virtual ~MyChild() {}
};

template <typename T> void virtualtest(T* t)
{
  static_assert(has_virtual_destructor<T>::value, "destructor must be virtual");
  delete t;
}
...
...
  MyParent* parent = new MyChild();
  virtualtest(parent);
...
```


EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

Delphi 2009の言語拡張とC++Builder 2009

staticプロパティ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- プロパティのGetter/Setterにstaticメソッドを利用
  - __propertyの宣言はstaticではないので注意してください
  - 「クラス名::プロパティ」または「インスタンス変数名.プロパティ」でアクセス可能

```
class MyStaticPropClass {
private:
  static int FX;
  static int GetX() { return FX; }
  static void SetX(int x) { FX = x; }
public:
  __property int X = {read = GetX, write = SetX };
};
int MyStaticPropClass::FX = 0;
...
...
  MyStaticPropClass::X = 1;
  int x = MyStaticPropClass::X;
  ...
  MyStaticPropClass c;
  c.X = 1;
  x = c.X;
...
```


おさらい

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- VCL/RTLのClasses::TListを利用する

```
static int __fastcall MyCompare(void* p1, void* p2)
{
  return *((int*)p1) - *((int*)p2);
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
  TList* list = new TList();
  for(int i = 0; i < 10; i++) {
    list->Add(new int(RandomRange(0, 99)));
  }
  list->Sort(MyCompare);
  for(int i = 0; i < list->Count; i++) {
    ListBox1->Items->Add(IntToStr(*((int*)list->Items[i])));
  }
  for(int i = 0; i < list->Count; i++) {
    delete (int*)list->Items[i];
    list->Items[i] = NULL;
  }
  delete list;
}
```

Generics::Collections::TList<T>

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 直接はC++Builderから利用できない
  - Delphiの.pasファイルで実装する
  - C++Builderのプロジェクトに「xxxx.pasファイル」を追加してビルドする
  - xxxx.hpp をインクルードし, xxxx.obj をリンクする

xxxx.pas

```
interface
uses Generics.Collections, Generics.Defaults;
type
  TMyList<T> = class
  protected
    list: TList<T>;
  public
    constructor Create;
    destructor Destroy; override;
    procedure Add(const Value: T);
    function Count: Integer;
    function GetItem(Index: Integer): T;
  end;

  MyComparer = class(TInterfacedObject, IComparer<Integer>)
    function Compare(const Left, Right: Integer): Integer;
  end;
  IntegerList = class(TMyList<Integer>)
    procedure Sort;
  end;
```


Delphiで実装し、C++Builderから利用する

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

xxxx.pas

```
implementation
constructor TMyList<T>.Create;
begin
  inherited;
  list := TList<T>.Create;
end;

destructor TMyList<T>.Destroy;
begin
  list.Free;
  inherited;
end;

procedure TMyList<T>.Add(const Value: T);
begin
  list.Add(Value);
end;

function TMyList<T>.Count: Integer;
begin
  Result := list.Count;
end;

function TMyList<T>.GetItem(Index: Integer): T;
begin
  Result := list[Index];
end;
```

```
function MyComparer.Compare(const Left, Right: Integer): Integer;
begin
  Result := (Left - Right);
end;

procedure IntegerList.Sort;
var
  comp: IComparer<Integer>;
begin
  comp := MyComparer.Create;
  list.Sort(comp);
end;
```

.cpp

```
#include "xxxx.hpp"

  IntegerList* list = new IntegerList();
  for(int i = 0; i < 10; i++) {
    list->Add(RandomRange(0, 99));
  }
  list->Sort();
  for(int i = 0; i < list->Count(); i++) {
    ListBox1->Items->Add(IntToStr(list->GetItem(i)));
  }
  delete list;
```

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

AnsiString, UTF8String, WideString, UnicodeString

AnsiString, UTF8String

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

- AnsiString, UTF8String はコードページ付き
  - dstring.h
    template <unsigned short CP>
    class AnsiStringT : public AnsiStringBase
- typedef AnsiStringT<0> AnsiString
  - sysmac.h
- typedef AnsiStringT<65001> UTF8String
  - System.hpp

WideString, UnicodeString

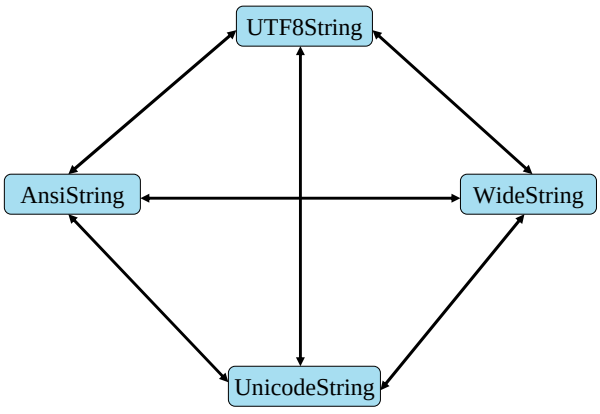
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- WideString
  - wstring.h
  - AnsiStringT<CP> を引数にとるコンストラクタ
  - WideChar(wchar_t) で管理
- UnicodeString
  - New!!
  - ustring.h
  - Delphi の UnicodeString と同じもの
  - wchar_t で管理

相互の変換

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

変換関数は不要になりました




```
AnsiString    ansi;
UTF8String    utf8;
WideString    wide;
UnicodeString uni;

ansi = utf8;
ansi = wide;
ansi = uni;

utf8 = ansi;
wide = ansi;
uni  = ansi;

utf8 = wide;
utf8 = uni;

wide = utf8;
uni  = utf8;

wide = uni;
uni  = wide;
```

文字列の初期化

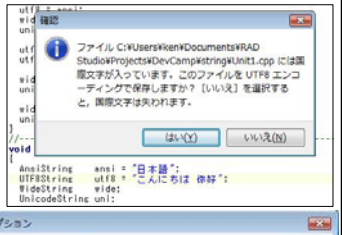
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

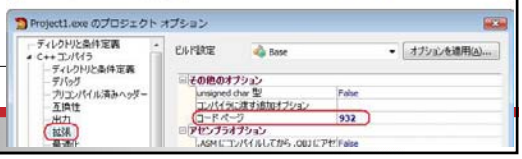
- “…” と L”…” と _TEXT(“…”) と _T(“…”)
  - tchar.h
  - _TCHAR（char または wchar_t）のマッピング
  - コンパイラのコードページ

```
#include <tchar.h>
…
  AnsiString       ansi   = "こんにちは";
  AnsiStringT<936> gb2312 = "你好";

  UTF8String    utf8 = L"こんにちは 你好";
  WideString    wide = L"こんにちは 你好";
  UnicodeString uni  = L"こんにちは 你好";

  // _TEXT() or _T()
  utf8 = _TEXT("こんにちは 你好");
  wide = _TEXT("こんにちは 你好");
  uni  = _TEXT("こんにちは 你好");
```





Quiz: マイグレーションとイベントハンドラ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

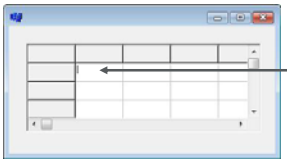
- 旧C++Builderで作成したプロジェクトを、C++Builder 2009 にインポートしました。
  - TStringGridを配置して、OnSetEditTextイベントを使用しています

```
class TForm1 : public TForm
{
__published:» // IDE 管理のコンポーネント
  TStringGrid *StringGrid1;
  void __fastcall StringGrid1SetEditText(TObject *Sender, int ACol, int ARow,
                                         const AnsiString Value);
private:» // ユーザー宣言
public:»» // ユーザー宣言
  __fastcall TForm1(TComponent* Owner);
};
```

```
void __fastcall TForm1::StringGrid1SetEditText(TObject *Sender, int ACol, int ARow,
                                               const AnsiString Value)
{
  this->Caption = Value;
}
```

UnicodeStringのハズですが…

  - 日本語/中国語混じりの文字列を入力してみます



テスト 你好

Quiz: マイグレーションとイベントハンドラ（続き）

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++Builder 2009 で ビルド & 実行しようとすると、どうなる？

  1. コンパイルエラーが発生する
  2. ビルドは成功するが、実行時にエラー/例外が発生する
  3. 日本語は正しく表示されるが、中国語が ？ に化ける
  4. フォント設定にも依るが、基本的に正しく表示される

_tmain, _tWinMain

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- プロジェクト新規作成で、main関数, WinMain関数 が変わります
- _TCHAR（char または wchar_t）のマッピングに依存します
  - main または wmain
  - WinMain または wWinMain

```
#include <tchar.h>
int _tmain(int argc, _TCHAR* argv[])

#include <tchar.h>
WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
```

- VCLのParamStr関数は、UnicodeStringを返します

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

IDEへの追加機能

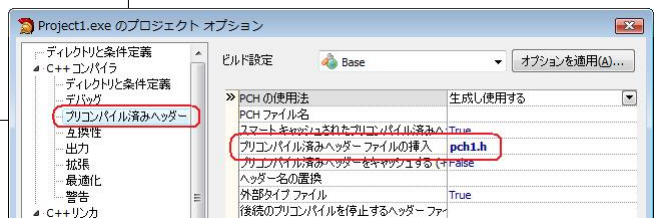
プリコンパイル済みヘッダーウィザード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- デフォルトで pch1.h を生成し、ビルド時に暗黙のうちにincludeされます
- #pragma hdrstop の上部は変更する必要なし

pch1.h

```
/*
...
...
*/
#ifndef pch1_H
#define pch1_H
#include <vcl.h>
#include <tchar.h>
#include <string>
#endif
```

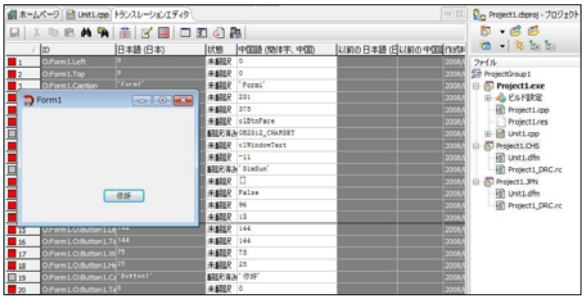


国際化支援機能 - ITE/ETM

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++Builder2006,C++Builder2007で消された
  - ITE（Integrated Translation Environment）
  - ETM（External Translation Manager）

  が**C++Builder2009**で**復活**しました!!

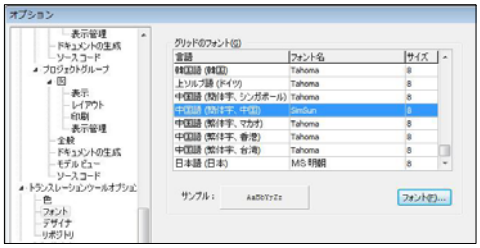


- 製品のインストール時に、英語版(English)を選択すると
  - VCL/RTLのシステムメッセージを、デフォルトで英語にすることも可能

開発手順のデモ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- IMEの設定を確認
- IDEの設定を確認
  - ITE/ETMで使用される言語別フォントの設定
- プロジェクトの新規作成
- 言語の追加
  - 日本語
  - 中国語
- .dfm および .RCの編集
  - 翻訳メモリ（リポジトリ）の利用



- テスト用のアクティブ言語の設定
  - レジストリに格納されます
    HKEY_CURRENT_USER¥Software¥CodeGear¥Locales

Tips: C++Builderでリソース文字列

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- .RCではなく、リソース文字列を XXXX.pas で定義する

```
unit Unit2;

interface

resourcestring rctest = 'rctest';

implementation

initialization
  rctest; // Delphiコンパイラによる最適化を防ぐ

end.
```

- VCLのLoadResourceString関数を利用する

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  ShowMessage(LoadResourceString(&Unit2::_rctest));
}
```

- ITE/ETMでも認識され、ローカライズ可能

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

Q&A