CODE GEAR FROM Borland Developer Camp

【A3】C++Builderテクニカルセッション

「C++Builder有効活用術」

CodeGear
エヴァンジェリスト
高橋智宏

CODE GEAR FROM Borland Developer Camp

アジェンダ

- C++Builderの良いところ
- 文字列型（VCLとSTL）
- VCLからSTLへ
- 巨大な整数の演算
- DLL遅延ロードオプションの意外な使いみち

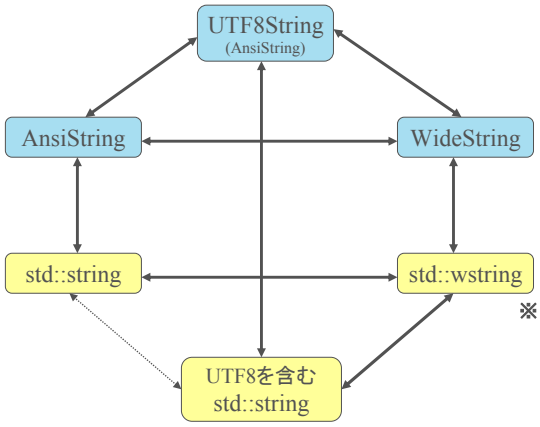
C++Builderの良いところ

- Delphiの.pasをコンパイルできる
- VCLが使える
- C/C++のランタイムが使える
- STL(Standard Template Library)が使える
- Delphi言語にはできないことが可能
  - Boost(スマートポインタなど)が使える
  - 演算子オーバーロード
  - リンカオプション(DLLの遅延ロード)

文字列型（VCLとSTL）

AnsiString
WideString
UTF8String
std::string
std::wstring

文字列型の相互変換



※wchar_t は環境依存
2バイトまたは4バイト

AnsiString, WideString, UTF8String

- AnsiString -> WideString
```
AnsiString ansi_str("abc");
WideString wide_str(ansi_str);
```
- WideString -> AnsiString
```
WideString wide_str(L"abc");
AnsiString ansi_str(wide_str);
```
- AnsiString -> UTF8String
```
AnsiString ansi_str("abc");
UTF8String utf8_str = AnsiToUtf8(ansi_str);
```
- UTF8String -> AnsiString
```
UTF8String utf8_str("abc");
AnsiString ansi_str = Utf8ToAnsi(utf8_str);
```
- WideString -> UTF8String
```
WideString wide_str(L"abc");
UTF8String utf8_str = UTF8Encode(wide_str);
```
- UTF8String -> WideString
```
UTF8String utf8_str(L"abc");
WideString wide_str = UTF8Decode(utf8_str);
```

VCL, STL

- AnsiString -> std::string

```
AnsiString ansi_str("abc");
string std_str(ansi_str.c_str());
```

- std::string -> AnsiString

```
string std_str("abc");
AnsiString ansi_str(std_str.c_str());
```

- WideString -> std::wstring

```
WideString wide_str(L"abc");
wstring std_wstr(wide_str.c_bstr());
```

- std::wstring -> WideString

```
wstring std_wstr(L"abc");
WideString wide_str(std_wstr.c_str());
```

- UTF8String -> std::string(UTF-8)

```
UTF8String utf8_str("abc");
string std_str(utf8_str.c_str());
```

- std::string(UTF-8) -> UTF8String

```
string std_str("abc");
UTF8String utf8_str(std_str.c_str());
```


std::string <-> std::wstring

- std::string -> std::wstring

```
setlocale(LC_ALL, "");
string std_str("abc");
int buf_size = _mbslen(std_str.c_str())+1;
wchar_t* buf = new wchar_t[buf_size];
memset(buf, 0, sizeof(wchar_t)*buf_size);
mbstowcs(buf, std_str.c_str(), std_str.length());
wstring std_wstr(buf);
delete [] buf;
```

- std::wstring -> std::string

```
setlocale(LC_ALL, "");
wstring std_wstr(L"abc");
int buf_size = (std_wstr.length()*MB_CUR_MAX)+1;
char* buf = new char[buf_size];
memset(buf, 0, sizeof(char)*buf_size);
wcstombs(buf, std_wstr.c_str(), std_wstr.length()*MB_CUR_MAX);
string std_str(buf);
delete [] buf;
```

std::wstring <-> std::string(UTF-8)

- Unicode, Inc. のサンプル実装を利用する
  - http://www.unicode.org/Public/PROGRAMS/CVTUTF/
    - ConvertUTF.h
    - ConvertUTF.c
- 最新のBoostを利用する
  - boost::to_utf8
  - boost::from_utf8
- Boostの utf8_codecvt_facet を利用する
  - http://www.javareading.com/bof/logs/2007/msg00145.html

VCLからSTLへ

複数の整数値を格納
ソートする
リストボックスに整数値を表示

VCL（TList）で書いてみる

```
TList* list = NULL;
try {
  list = new TList();
  for(int i = 0; i < 10; i++) {
    list->Add(new int(RandomRange(0, 99)));
  }
  list->Sort(MyCompare2);
  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;
  }
}
__finally {
  if( list != NULL ) {
    delete list;
    list = NULL;
  }
}
```

```
static int __fastcall MyCompare1(void* p1, void* p2) {
  return *((int*)p1) - *((int*)p2);
}
static int __fastcall MyCompare2(void* p1, void* p2) {
  return *((int*)p2) - *((int*)p1);
}
```


new int[…], qsort関数 に替えてみる

```
int* ia = NULL;
try {
  int ia_size = 10;
  ia = new int[ia_size];
  for(int i = 0; i < ia_size; i++) {
    ia[i] = rand()%100;
  }
  qsort(ia, ia_size, sizeof(*ia), QSortCompare2);
  for(int i = 0; i < ia_size; i++) {
    ListBox1->Items->Add(IntToStr(ia[i]));
  }
}
__finally {
  if( ia != NULL ) {
    delete [] ia;
    ia = NULL;
  }
}
```

```
static int QSortCompare1(const void *e1, const void *e2) {
  return *((int*)e1) - *((int*)e2);
}
static int QSortCompare2(const void *e1, const void *e2) {
  return *((int*)e2) - *((int*)e1);
}
```

スマートポインタ(Boost)に替えてみる

```
int ia_size = 10;
boost::shared_ptr<int> ia(new int[ia_size]);
for(int i = 0; i < ia_size; i++) {
  ia.get()[i] = rand()%100;
}
qsort(ia.get(), ia_size, sizeof(*ia), QSortCompare2);
for(int i = 0; i < ia_size; i++) {
  ListBox1->Items->Add(IntToStr(ia.get()[i]));
}
```


```
static int QSortCompare1(const void *e1, const void *e2) {
  return *((int*)e1) - *((int*)e2);
}
static int QSortCompare2(const void *e1, const void *e2) {
  return *((int*)e2) - *((int*)e1);
}
```

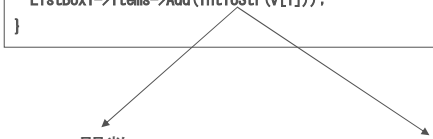

scoped_array に替えてみる

```
int ia_size = 10;
boost::scoped_array<int> ia(new int[ia_size]);
for(int i = 0; i < ia_size; i++) {
  ia[i] = rand()%100;
}
qsort(ia.get(), ia_size, sizeof(*(ia.get())), QSortCompare2);
for(int i = 0; i < ia_size; i++) {
  ListBox1->Items->Add(IntToStr(ia[i]));
}
```

```
static int QSortCompare1(const void *e1, const void *e2) {
  return *((int*)e1) - *((int*)e2);
}
static int QSortCompare2(const void *e1, const void *e2) {
  return *((int*)e2) - *((int*)e1);
}
```

STL（vector, sort）に替えてみる

```
vector<int> v;
for(int i = 0; i < 10; i++) {
  v.push_back(rand()%100);
}
sort(v.begin(), v.end(), greater<int>()); // or less<int>()
for(int i = 0; i < v.size(); i++) {
  ListBox1->Items->Add(IntToStr(v[i]));
}
```



itoa関数

```
char temp[25];
itoa(v[i], temp, 10);
ListBox1->Items->Add(AnsiString(temp));
```

ostringstream

```
ostringstream os;
os << v[i];
string temp = os.str();
ListBox1->Items->Add(AnsiString(temp.c_str()));
```


巨大な整数の演算

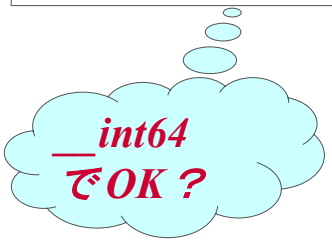
65536 の 8乗 は？
A = 65536
B = A x A x A x A x A x A x A x A

VCL（FMTBcd, 2進化10進数）を使ってみる

```
TBcd a = IntegerToBcd(65536);
TBcd b, c;
BcdMultiply(a, a, b);
BcdMultiply(b, a, c);
BcdMultiply(c, a, b);
BcdMultiply(b, a, c);
BcdMultiply(c, a, b);
BcdMultiply(b, a, c);
BcdMultiply(c, a, b);
AnsiString msg = BcdToStr(b);
ShowMessage(msg);
```


__int64 に替えてみる

```
__int64 a = 65536i64;
__int64 b = (a * a * a * a * a * a * a * a);
AnsiString msg = IntToStr(b);
ShowMessage(msg);
```



C++ Big Integer Library を使ってみる

オープンソース（public domain）
http://mattmccutchen.net/bigint/

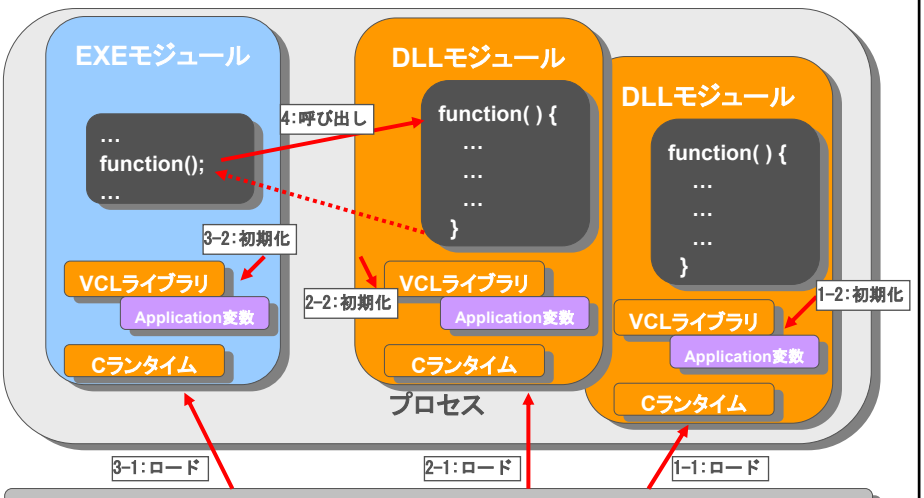

```
#include "BigIntegerLibrary.hh"
...
BigInteger a = 65536;
BigInteger b = (a * a * a * a * a * a * a * a);
std::string str = easyBItoString(b);
AnsiString msg(str.c_str());
ShowMessage(msg);
```


DLLの遅延ロード

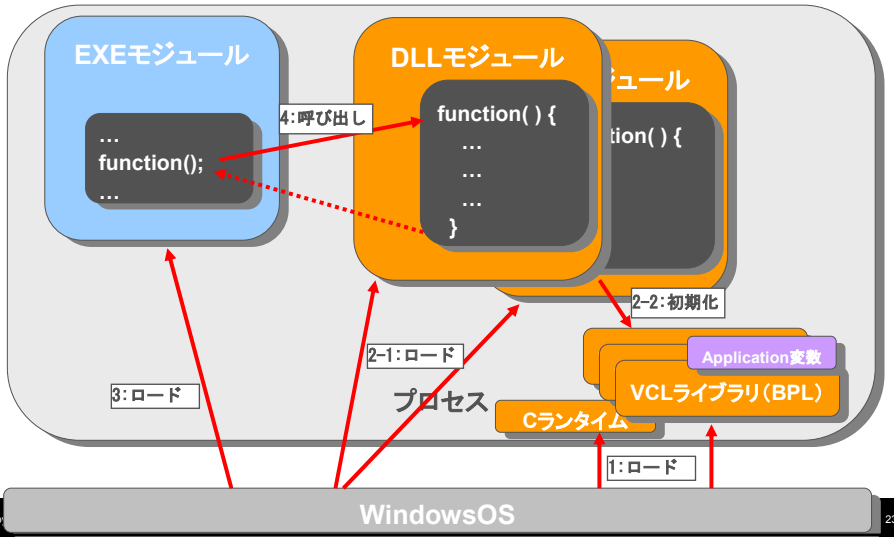
DLLからフォーム(Form)を表示
VCLグローバル変数の初期化問題
Delphiには無いC++Builderの機能

典型的な問題

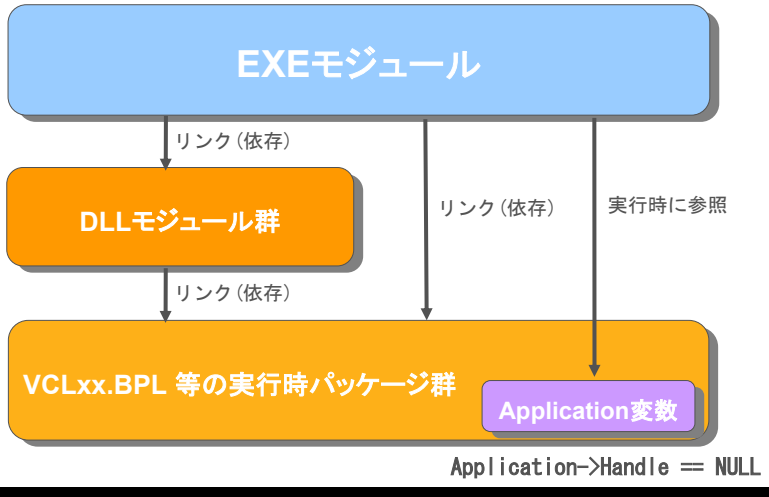
よくある構成 – その1



よくある構成 – その2



問題点



原因

- 原因は、EXEファイル実行時におけるBPLの初期化の過程にある
- WindowsOSがEXEモジュールをロードする際には、リンクされている(依存している)DLLモジュールから先にロードされる
- ロードされようとしているモジュールが、他のBPLやDLLをさらにリンクしている場合、それらBPL,DLLが先にロードされる
- ロードされたEXEモジュールまたはDLLモジュールは、BPLのグローバル変数の初期化を行うよう、いったんBPLに制御を移す

原因（続き）

- 特定のBPLには、VCLグローバル変数が定義されており、BPLのロード後に一度だけ初期化されるものがある
  - 例えば、Application変数は Formsユニットのグローバル変数であり、VCLxx.BPL のロード後に一度だけ初期化される
- BPLのソースコード中には、Boolean型のIsLibraryというグローバル変数が存在し、EXE,DLLロード時に書き換わる
  - DLLモジュールがロードされる際、IsLibraryはTrueにセットされる
  - EXEモジュールがロードされる際、IsLibraryはFalseにセットされる
- BPLのソースコード中には、IsLibrary変数を参照しつつ、BPLの初期化を行うコード等が存在し、IsLibraryがFalseの場合のみ開発者が期待する動作を行うものがある

原因（続き） ー IsLibrary変数の使用例

- Application変数は、VCLxx.BPL内にあるControlsユニットの初期化コード内で生成される
  - Application := TApplication.Create(nil);
  - 具体的にはcontrols.pas最下部のinitializationブロック
- initializationブロックは、EXEまたはDLLにより一度だけコールされる
- ApplicationのHandleプロパティは、TApplicationクラスのコンストラクタ内でCreateHandleが呼び出されることにより初期化されるべきであるが、CreateHandleは、IsLibrary が Flase の場合のみ呼び出される（仕様）
  - if not IsLibrary then CreateHandle;
  - 具体的にはforms.pas内のTApplication.Createメソッド内
- DLLモジュールがロードされ、DLLモジュールによりVCLxx.BPLが初期化される場合、IsLibraryはTrueである

VCL（BPL）初期化問題の解決方法

- DLLが本当に必要となる時点までロードされず、EXEファイルのロード時にBPLの初期化が行われる必要がある
  - DLLのロードを最大限遅らせる手法として、LoadLibrary API, GetProcAddress APIを使用して、DLLがexportしている関数のアドレスを動的に取得し、その関数を呼び出すものがある
  - LoadLibrary API, GetProcAddress API を利用することによる欠点として、function(...); のような関数呼び出しのコードを直接書くことが出来ず、ソースコードが乱雑になり、工数が増える
- そこで、C++Builder の「リンカ(詳細)」オプションとして新設された「遅延ロード」を使用し、EXEが使用するDLLを指定して再構築することにより、リンカが LoadLibrary API, GetProcAddress API の呼び出しコードを自動生成してくれる

※マルチスレッドにも対応

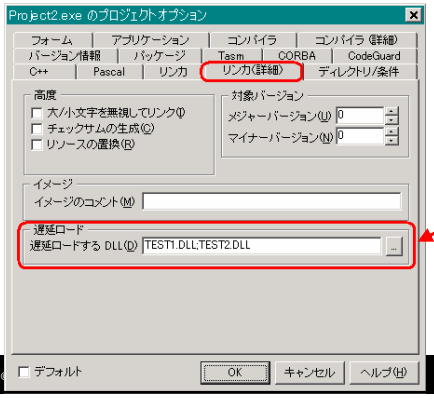
VCL（BPL）初期化問題の解決方法（続き）

- C++Builderの「遅延ロード」オプションを使用すると、function(...); のような一般的な関数の呼び出し方を可能にしつつ、関数の最初の呼び出し時にLoadLibrary API,GetProcAddress APIを呼び出すフックルーチンが呼び出され、その後本来の関数が呼び出される。
- フックルーチンは具体的には、Source¥Rtl¥Source¥misc¥delayhlp.c の __delyLoadHelper関数および、スタブコード
- 2回目以降の関数呼び出し時にはフックルーチンは呼び出されないよう、内部のジャンプテーブルが書き換わっており、直接本来の関数への呼び出しが行われる
  - 「Visual C++」の実装方法とは随分異なるものの、根底にある仕組みは同じ

（※Microsoft Systems Journal誌 Dec 1998 Vol.13 Matt Pietrek氏の記事を参照）

VCL（BPL）初期化問題の解決方法（続き）

- リンカ(ilink32.exe)オプション -d にDLL名を指定する
  - 例: … -dTEST1.DLL -dTEST2.DLL …
  - PATHを指定するのではないことに注意



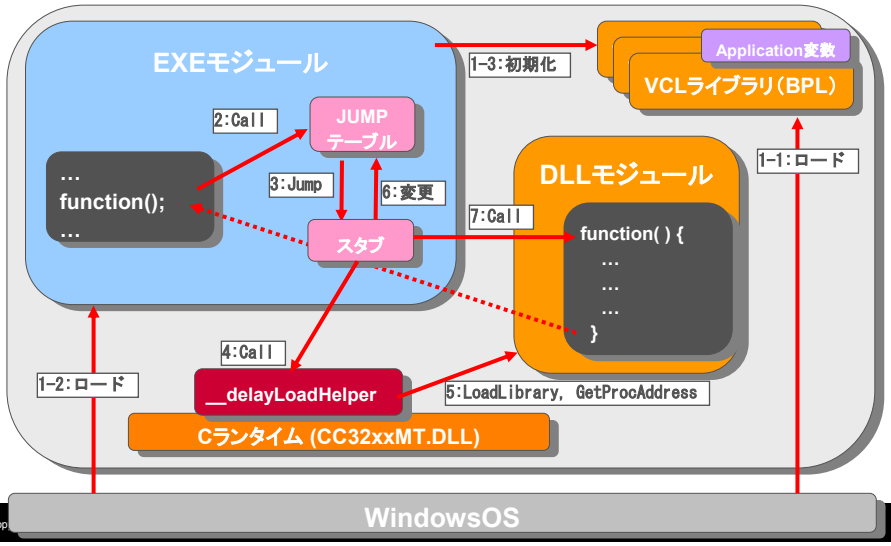
設定ダイアログではセミコロン;で区切る

例　TEST1.DLL;TEST2.DLL

※kernel32.dll等はもちろん×

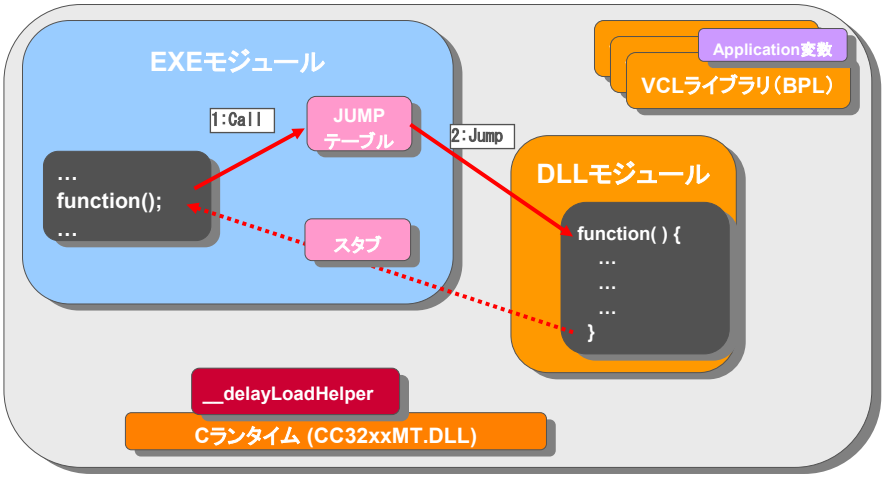
CODE GEAR FROM Borland Developer Camp

遅延ロード時のDLLのリンクと呼び出し（1回目）



CODE GEAR FROM Borland Developer Camp

遅延ロード時のDLLのリンクと呼び出し（2回目以降）



その他の不完全な解決策

- 「よくある構成 – その1」のように、すべてのEXE,DLLにVCL（BPL）をスタティックリンクする
- Application->Initialize(); の直後に

  Application->CreateHandle();

  を呼び出す
- C++Builder 2007 で Application->Initialize(); の直後に

  SetApplicationMainFormOnTaskBar(Application, true);

  を呼び出す

サンプルプロジェクトによる検証