【B4】C++テクニカルセッション

EMBARCADERO TECHNOLOGIES

DEVELOPER CAMP

C++Builder ライブラリ徹底活用法

株式会社　日本情報システム
筑木真志

アジェンダ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++における標準的ライブラリひと巡り
- 1時間で作る(?)簡易テキストファイル検索ツール
  - ファイルのワイルドカード検索
  - 文字コードの自動判別
  - boost::regex（正規表現クラス）の使い方
- まとめ

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

C++における標準的ライブラリひと巡り

C++における標準的ライブラリの位置づけ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 標準C++ライブラリ
  - 入出力(ストリーム) 、文字列、数値処理、例外など
  - STL(Standard Template Library)
  - 標準Cライブラリ
- Boost C++ Libraries
  - 標準C++ライブラリを補完するオープンソースのライブラリ群
  - 商用／非商用関係なく自由に使用可能
  - C++標準化委員会のメンバーが開発に参加
  - 一部のライブラリは、C++0xで標準C++ライブラリに取り込まれる
    - Technical Report 1(TR1)と呼ばれる

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

STLとは

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 一般的なロジックをテンプレートを使用して実装したライブラリ

```
#include <string>
#include <iostream>
#include <vector>
#include <algorithm>
#include <tchar.h>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  vector<string> v;
  vector<string>::iterator i;

  v.push_back(_T("Alain Prost"));
  v.push_back(_T("Nelson Piquet"));
  v.push_back(_T("Ayrton Senna"));
  v.push_back(_T("Nigel Mansell"));
  v.push_back(_T("Michael Schumacher"));
  v.push_back(_T("Damon Hill"));
  v.push_back(_T("Jacques Villeneuve"));
  v.push_back(_T("Mika Häkkinen"));
  v.push_back(_T("Fernando Alonso"));
  v.push_back(_T("Kimi Räikkönen"));
  v.push_back(_T("Lewis Hamilton"));

  sort(v.begin(), v.end());

  for (i = v.begin(); i != v.end(); ++i) {
    cout << *i << endl;
  }

  return 0;
}
```

可変長配列

値の追加

ソート

値の列挙

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <tchar.h>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  vector<int> v;
  vector<int>::iterator i;

  v.push_back(61);
  v.push_back(25);
  v.push_back(38);
  v.push_back(51);
  v.push_back(94);
  v.push_back(98);
  v.push_back(11);
  v.push_back(29);
  v.push_back(58);
  v.push_back(79);
  v.push_back(23);

  sort(v.begin(), v.end());

  for (i = v.begin(); i != v.end(); ++i) {
    cout << *i << endl;
  }

  return 0;
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

STLの構成要素

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- コンテナ
  - 値の集合(可変長配列、双方向リスト、連想配列など)
- イテレータ(反復子/列挙子)
  - コンテナの中身を指し示すもの≒ポインタ
- アルゴリズム
  - 探索、ソート、除外など、コンテナに対する一般的なアルゴリズム
- 関数オブジェクト(ファンクタ)
  - 値に対する操作(比較、演算など)

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

STLコンテナ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 順序コンテナ
  - vector/deque(可変長配列)
  - list(双方向リスト)
- 連想コンテナ
  - map/multi_map(キーと値の組み合わせ:連想配列)
  - set/multi_set(値がキーを兼ねるオブジェクトの集合)
- コンテナアダプタ
  - stack(LILO待ち行列)
  - queue(FILO待ち行列)
  - priority_queue(優先度つき待ち行列)

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

コンテナの例

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
#include <string>
#include <iostream>
#include <vector>
#include <algorithm>
#include <tchar.h>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  vector<string> v;
  vector<string>::iterator i;

  v.push_back(_T("Alain Prost"));
  v.push_back(_T("Nelson Piquet"));
  v.push_back(_T("Ayrton Senna"));
  v.push_back(_T("Nigel Mansell"));
  v.push_back(_T("Michael Schumacher"));
  v.push_back(_T("Damon Hill"));
  v.push_back(_T("Jacques Villeneuve"));
  v.push_back(_T("Mika Häkkinen"));
  v.push_back(_T("Fernando Alonso"));
  v.push_back(_T("Kimi Räikkönen"));
  v.push_back(_T("Lewis Hamilton"));

  sort(v.begin(), v.end());

  for (i = v.begin(); i != v.end(); ++i) {
    cout << *i << endl;
  }

  return 0;
}
```

ヘッダファイルのインクルード:
vector型を使用するならば、vectorをインクルード
コンテナ型の名称とヘッダファイル名は一致する

コンテナの宣言:
string型を持つ可変長配列(vector)
コンテナにしたいクラスを<>でくくる

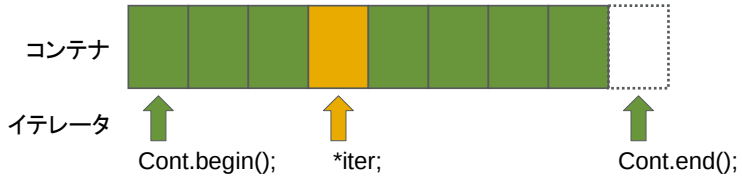
コンテナへの要素追加:
push_back()メソッド:コンテナの末尾に要素を追加

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

イテレータ(反復子/列挙子)

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- イテレータとは、コンテナに特化したポインタ



- ポインタとの違い
  - コンテナの「先頭」と「末尾」を指す特殊な値が存在
    - begin()メソッド、end()メソッド
  - コンテナによって演算に制限がある場合がある
    - インクリメント、デクリメント、ランダムアクセス

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

イテレータの例

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
#include <string>
#include <iostream>
#include <vector>
#include <algorithm>
#include <tchar.h>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  vector<string> v;
  vector<string>::iterator i;

  v.push_back(_T("Alain Prost"));
  v.push_back(_T("Nelson Piquet"));
  v.push_back(_T("Ayrton Senna"));
  v.push_back(_T("Nigel Mansell"));
  v.push_back(_T("Michael Schumacher"));
  v.push_back(_T("Damon Hill"));
  v.push_back(_T("Jacques Villeneuve"));
  v.push_back(_T("Mika Häkkinen"));
  v.push_back(_T("Fernando Alonso"));
  v.push_back(_T("Kimi Räikkönen"));
  v.push_back(_T("Lewis Hamilton"));

  sort(v.begin(), v.end());

  for (i = v.begin(); i != v.end(); ++i) {
    cout << *i << endl;
  }

  return 0;
}
```

イテレータの宣言:
container<type>::iterator i;
コンテナには必ずiterator型が定義されている

コンテナの先頭と末尾
begin()メソッド:コンテナの先頭を指すイテレータ を返す
end ()メソッド:コンテナの末尾を指すイテレータ を返す

注意:コンテナの末尾とは最後の要素の「次」である

イテレータの演算:
インクリメントでイテレータは次の要素を指す

要素へのアクセス:
ポインタ同様、＊演算子で要素にアクセス

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

STLの主なアルゴリズム

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ヘッダファイル
  - ヘッダファイルalgorithmをインクルード
- 検索・走査アルゴリズム
  - find/find_if　検索
  - for_each　列挙
- 配列操作アルゴリズム
  - copy　要素の複写
  - remove/remove_if　要素の削除
  - replace/replace_if　要素の置換
- ソート関連アルゴリズム
  - sort　ソート
  - nth_element　n番目の要素

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Boost C++ Libraries概要

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 入手
  - http://www.boost.org/でソースコード・ドキュメントが入手可
  - C++Builder 2009にVersion 1.35.0がバンドル
  - 最新版はVersion1.38.0
- 主なライブラリ
  - 文字列・テキスト処理 (トークン分割、正規表現など)
  - スマートポインタ (参照カウンタ付きポインタ、改良型auto_ptr)
  - マルチスレッド
  - グラフ理論 (双方向グラフ、経路探索)
  - 無名関数(ラムダ式)
  - シリアライズ（直列化）
  - など

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Boost C++ Libraries概要

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++Builder 2009で動作する主なライブラリ
  - format　printf関数のような書式化
  - string_algo　文字列処理
  - regex　正規表現パーサ
  - tokenizer　トークン分割
  - shared_ptr　参照カウンタ付きポインタ
  - tuple　n個組値(≒メンバ名が無い構造体)
  - filesystem　ファイル・ディレクトリ操作
  - array　コンテナのように振る舞う固定長配列

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1時間で作る(?)簡易テキストファイル検索ツール
Step0：前準備

要求仕様

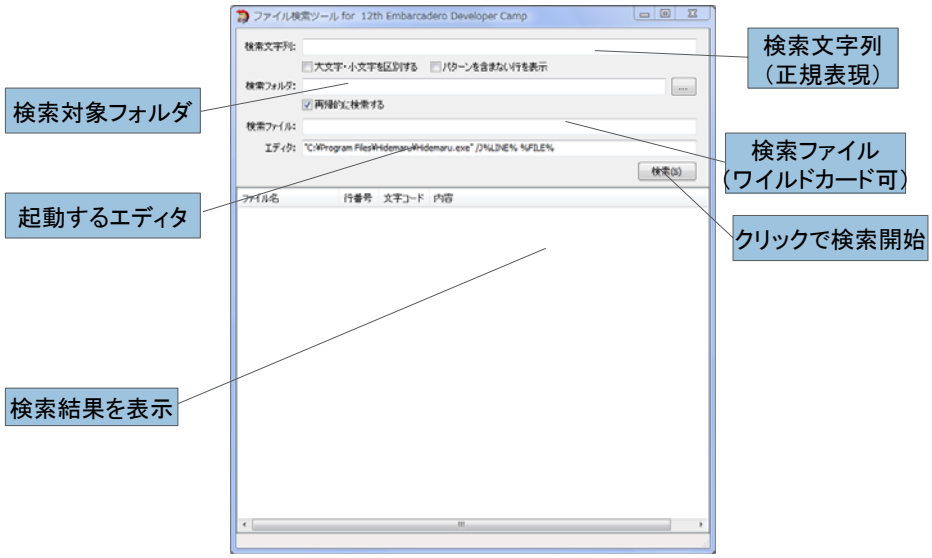
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 正規表現を使用して、任意の文字列を含むファイルを検索する
- 検索元は任意のフォルダを指定できること
- 検索対象ファイルはワイルドカードで複数種類指定できる
- ファイルのエンコーディングは自動識別
  - Unicode(UTF-7/UTF-8/UTF-16)
  - 日本語(Shift-JIS/EUC-JP/JIS)
- 検索結果から任意のテキストエディタを起動できること

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

画面説明

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

処理手順

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1. 対象ファイルを探す
2. 見つかったファイルのエンコーディングを調べる
3. ファイル中身をUnicodeに変換しながらバッファに取り込む
4. 行単位で正規表現マッチングを行う
5. 結果をTListViewに出力

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1時間で作る(?)簡易テキストファイル検索ツール
Step1：ファイルのワイルドカード検索

ファイルの検索を行うには

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- VCLの場合
  - FindFirst / FindNext / FindClose関数
- JVCLの場合
  - TJvSearchFilesコンポーネント
- boostの場合
  - boost::filesystem

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
int __fastcall TfrmMain::FindFiles(const UnicodeString& Path,
  const UnicodeString& Filter, bool FindSubFolder, const boost::wregex& reg)
{
  // 拡張子を区切り文字でトークン分割
  std::unique_ptr<TStringList> pFilterList(new TStringList());
  pFilterList->Delimiter = L';';
  pFilterList->DelimitedText = Filter;

  // パス区切り文字で終わることを保証
  UnicodeString Folder = IncludeTrailingPathDelimiter(Path);

  int Count = FindFilesSub(Folder, pFilterList.get(), FindSubFolder, reg);

  return Count;
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
int __fastcall TfrmMain::FindFilesSub(const UnicodeString& Path, TStrings* pFilter, bool FindSubFolder)
{
  TSearchRec sr;

  // ファイルの列挙
  for (int i = 0; i < pFilter->Count; ++i) {
    if (FindFirst(Path + pFilter->Strings[i], faAnyFile, sr) == 0) {
      Application->ProcessMessages();
      do {
        lvwFiles->Items->Add()->Caption = sr.Name;
      } while (FindNext(sr) == 0);
      FindClose(sr);
    }
  }

  if (FindSubFolder) {
    if (FindFirst(Path + _T("*.*"), faDirectory, sr) == 0) {
      Application->ProcessMessages();
      do {
        if (sr.Attr & faDirectory) {
          if (sr.Name == _T(".") || sr.Name == _T("..")) continue;
          // サブフォルダを検索
          UnicodeString SubPath = IncludeTrailingPathDelimiter(Path + sr.Name);
          FindFilesSub(SubPath, pFilter, FindSubFolder, reg);
        }
      } while (FindNext(sr) == 0);
      FindClose(sr);
    }
  }
  return lvwFiles->Items->Count;
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

新しいスマートポインタ: unique_ptr

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++0xでauto_ptrに代わりunique_ptrが導入
- auto_ptrは非推奨(deprecated)となる
- 使い勝手はauto_ptrと同一

```
#include <memory>

int _tmain(int argc, _TCHAR* argv[])
{
  std::unique_ptr<TStringList> pList0(new TStringList());

  pList0->Delimiter = ';';
  pList0->DelimitedText = _T("*.cpp;*.cxx;*.cc;*.c;*.hpp;*.hxx;*.hh;*.h");

  UnicodeString us;
  us.printf(_T("Count = %d"),pList0->Count);
  ShowMessage(us);

  return 0;
}
```

- “auto_ptr”と記述していた部分を、“unique_ptr”と書き換えるだけ

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

新しいスマートポインタ: unique_ptr

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- unique_ptr間の代入はコンパイラによって抑止される

```
std::unique_ptr<TStringList> pList0(new TStringList());
std::unique_ptr<TStringList> pList1 = pList0;
```

コンパイルしてみると…

[BCC32 エラー] File1.cpp(13): E2247 'std::unique_ptr<TStringList,std::default_delete<TStringList>>::unique_ptr(const std::unique_ptr<TStringList,std::default_delete<TStringList> > &)' はアクセスできない

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

新しいスマートポインタ: unique_ptr

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ポインタの所有権を移動するには…
  - C++0xで導入された「ムーブセマンティクス(所有権の移動)」を使用する

```
std::unique_ptr<TStringList> pList0(new TStringList());
std::unique_ptr<TStringList> pList1 = std::move(pList0);
```

- unique_ptrが保持しているポインタを取得するには…
  - unique_ptr::get()関数を使用する

```
std::unique_ptr<TStringList> pFilterList(new TStringList());
pFilterList->Delimiter = L';';
pFilterList->DelimitedText = Filter;

int Count = FindFilesSub(Folder, pFilterList.get(), FindSubFolder, reg);
```

  - get関数の戻り値がNULLならば、そのunique_ptrはポインタを保持していない。

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1時間で作る(?)簡易テキストファイル検索ツール

Step2 : テキストファイルの文字コード判別

テキストファイルとは？

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 一般的に、「人間」が「読む」ことが出来るファイル
  - 文字コード（符号化方式）
    - Unicode(UTF-7,UTF-8,UTF-16,UTF-32など…)
    - ISO-2022-JP (JISコード)
    - Shift-JIS
    - 日本語EUC
    - など…
  - 改行コード
    - CR/LF (DOS/Windows)
    - LF (*nix系)
    - CR (旧MacOS)

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

文字コードの判別方法

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ファイルがUnicodeであると判っている場合
  - BOM (Byte Order Mark)の並びをチェックする
  - BOMが無いUTF-8なテキストファイルも存在する
    → ASCII文字のみで構成されたファイルと区別がつかない
  - UTF-16BEとUTF-16LEはBOMをもたないため判定が難しい
    → 一般的にUTF-16BEとUTF-16LEの使用は非推奨
- 非Unicodeの場合
  - ファイルに出現するコードの傾向より推定する。
  - それでも、正確に判定できるとは言えない

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Mlangを使った文字コードの判別

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Mlang(mlang.dll)はWindows標準の多言語処理ライブラリ
- COMで実装されているのでCoInitialize/CoUninitializeが必要

```
WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
{

  CoInitialize(NULL); // COMの初期化

  try {
    Application->Initialize();
    Application->MainFormOnTaskBar = true;
    Application->CreateForm(__classid(TfrmMain), &frmMain);
    Application->Run();
  } catch (Exception &exception) {
    Application->ShowException(&exception);
  } catch (...) {
    try {
      throw Exception("");
    } catch (Exception &exception) {
      Application->ShowException(&exception);
    }
  }

  CoUninitialize(); // COMの終了処理

  return 0;
}
```

最初にCoInitialize

最後にCoUninitialize

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

MIangを使ったエンコーディングの判別

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- IMultiLanguage2インターフェースの使い方
  1. mlang.hをインクルードする
  2. CoCreateInstanceでIMultiLanguage2インターフェースを生成

```
// IMultiLanguage2の生成
m_pMLang = NULL;
HRESULT result = CoCreateInstance(CLSID_CMultiLanguage,
  NULL, CLSCTX_INPROC_SERVER, IID_IMultiLanguage2, (void**)&m_pMLang);
if (FAILED(result)) m_pMLang = NULL;
```

  3. IMultiLanguage2::DetectInputCodepageを使って文字コードを判別

```
//文字コードを判別する
int Encodings = 1;
DetectEncodingInfo EncodingInfo[1];
::ZeroMemory(EncodingInfo, sizeof(EncodingInfo));

HRESULT result = m_pMLang->DetectInputCodepage(0, 0,
    pBuffer, &BufferSize, EncodingInfo, &Encodings);
if (FAILED(result)) return 0;
return EncodingInfo[0].nCodePage;
```

  4. IMultiLanguage2::Release()を呼び出してメモリを解放する

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.h

```
//---------------------------------------------------------------------------
#ifndef DetectCodepageH
#define DetectCodepageH
//---------------------------------------------------------------------------
#include <vcl.h>
#include <mlang.h>

class CDetectCodepage
{
public:
  CDetectCodepage();
  virtual ~CDetectCodepage();

  int DetectCodepage(CHAR* pBuffer, int BufferSize); // コードページ識別
  UnicodeString GetCodePageName(int CodePage); // コードページ名取得

private:
  IMultiLanguage2* m_pMLang;
};

#endif
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <mlang.h>
#pragma hdrstop
#include <tchar.h>

#include "DetectCodepage.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)
CDetectCodepage::CDetectCodepage() :
  m_pMLang(NULL)
{
  // IMultiLanguage2の生成
  HRESULT result = CoCreateInstance(CLSID_CMultiLanguage,
    NULL, CLSCTX_INPROC_SERVER, IID_IMultiLanguage2, (void**)&m_pMLang);

  if (FAILED(result)) m_pMLang = NULL;
}

CDetectCodepage::~CDetectCodepage()
{
  if (m_pMLang) m_pMLang->Release();
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.cpp（続き）

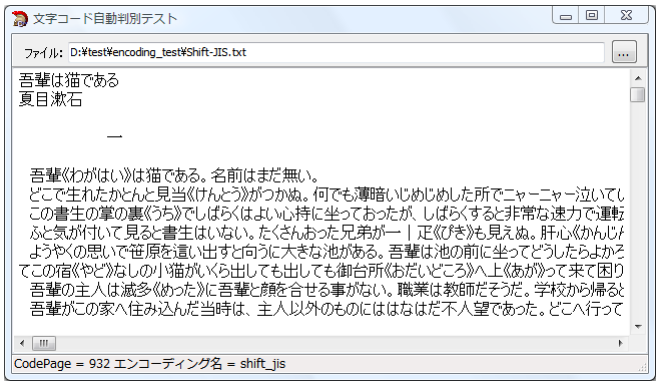
```
int CDetectCodepage::DetectCodepage(CHAR* pBuffer, int BufferSize)
{
  if (m_pMLang == NULL) return -1;

  //文字コードを判別する
  int Encodings = 1;
  DetectEncodingInfo EncodingInfo[1];
  ::ZeroMemory(EncodingInfo, sizeof(EncodingInfo));

  HRESULT result = m_pMLang->DetectInputCodepage(0, 0, pBuffer, &BufferSize, EncodingInfo, &Encodings);
  if (FAILED(result)) return 0;
  return EncodingInfo[0].nCodePage;
}

UnicodeString CDetectCodepage::GetCodePageName(int CodePage)
{
  UnicodeString CodePageName(_T("不明"));
  if (m_pMLang) {
    MIMECPINFO CodePageInfo;
    LANGID LangID = PRIMARYLANGID(GetSystemDefaultLangID());
    HRESULT result = m_pMLang->GetCodePageInfo(CodePage, LangID, &CodePageInfo);
    if (SUCCEEDED(result)) {
       CodePageName = UnicodeString(CodePageInfo.wszWebCharset);
    }
  }
  return CodePageName;
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

エンコーディングの判別をテストする

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- テキストファイルの文字コード自動判別



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- コードページ番号を元にファイルを読み込む

```
void __fastcall TfrmMain::loadFile(const UnicodeString& FileName)
{
  LabeledEdit1->Text = FileName;

  static CDetectCodepage Detecter;

  std::unique_ptr<TStringStream> pStream(new TStringStream());
  pStream->LoadFromFile(FileName);

  CHAR* pBuffer = &pStream->Bytes[0];
  int BufferSize = std::min(1024i64, pStream->Size);

  // コードページを元にファイルを読み込む
  int CodePage = Detecter.DetectCodepage(pBuffer, BufferSize);
  TEncoding* pEncoding = TEncoding::GetEncoding(CodePage);
  Memo1->Lines->LoadFromStream(pStream.get(), pEncoding);

  // TEncoding::GetEncodingで取得したエンコード情報は
  // delete で解放する必要がある
  delete pEncoding;

  // エンコーディング情報を取得
  UnicodeString CodePageName;
  CodePageName = Detecter.GetCodePageName(CodePage);

  UnicodeString us;
  us.printf(_T("CodePage = %d エンコーディング名 = %ls"), CodePage, CodePageName.t_str());
  StatusBar1->SimpleText = us;
}
```

オブジェクトの解放はdeleteで行う。
System::TObject::Free()で解放してはならない。

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

コードページ判定での問題点：その1

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- TEncoding::GetEncodingでエラーが発生する
  - CP51932(EUC-JP)の場合：
    - Windows上でEUC-JPとされるコードページはCP20932とCP51932
    - CP20932 とCP51932の間に互換性がない（互いに欠落部分がある）
    - VCLでCP51932を扱う場合は、CP20932で代用する
  - CP1200(UTF-16)の場合：
    - TEncoding::UnicodeをTStrings::LoadFromStreamへ渡す
  - CP1201(UTF-16BE)
    - TEncoding::BigEndianUnicodeをTStrings::LoadFromStreamへ渡す

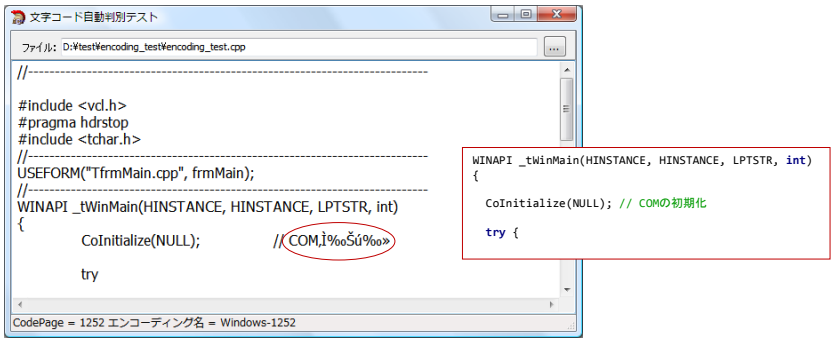
本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

コードページ判定での問題点：その2

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- CP932(Shift-JIS)のはずが、CP932と判定されない

  例：CP1252(Western)、CP20127(US-ASCII)



  - 対応策として…
    - IMultiLanguage2::DetectInputCodepageの候補数を増やす
    - 取り扱う文字コードを限定する

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.h

```
//---------------------------------------------------------------------------
#ifndef DetectCodepageH
#define DetectCodepageH
//---------------------------------------------------------------------------
#include <vcl.h>
#include <mlang.h>
#include <set>

class CDetectCodepage
{
public:
  CDetectCodepage();
  virtual ~CDetectCodepage();

  int DetectCodepage(CHAR* pBuffer, int BufferSize); // コードページ識別
  UnicodeString GetCodePageName(int CodePage); // コードページ名取得

private:
  IMultiLanguage2* m_pMLang;
  int m_AnsiCodePage; // Ansiコードページ(システムのコードページ)
  std::set<int> m_DetectCP; // 有効なコードページ
};

#endif
```

STLのsetを使用して
有効なコードページを限定する

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.cpp(修正部のみ)

```
CDetectCodepage::CDetectCodepage() :
    m_pMLang(NULL), m_AnsiCodePage(GetACP())
{
    // IMultiLanguage2の生成
    m_pMLang = NULL;
    HRESULT result = CoCreateInstance(CLSID_CMultiLanguage,
        NULL, CLSCTX_INPROC_SERVER, IID_IMultiLanguage2, (void**)&m_pMLang);
    if (FAILED(result)) m_pMLang = NULL;

    // 有効なコードページ
    m_DetectCP.insert(932);    // Shift-JIS
    m_DetectCP.insert(51932);  // EUC-JP
    m_DetectCP.insert(20932);  // 日本語 (JIS 0208-1990 and 0212-1990)
    m_DetectCP.insert(50220);  // 日本語 (JIS)
    m_DetectCP.insert(1200);   // UTF-16
    m_DetectCP.insert(1201);   // UTF-16(BE)
    m_DetectCP.insert(65000);  // UTF-7
    m_DetectCP.insert(65001);  // UTF-8
}
```

デフォルトのコードページを取得しておく

処理対象となるコードページをSTLコンテナ(set)に保存

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

サンプルコード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- DetectCodepage.cpp（修正部のみ）

```
int CDetectCodepage::DetectCodepage(CHAR* pBuffer, int BufferSize)
{
  if (m_pMLang == NULL) return m_AnsiCodePage;

  //文字コードを判別する
  int Encodings = 10;
  DetectEncodingInfo EncodingInfo[10];
  ::ZeroMemory(EncodingInfo, sizeof(EncodingInfo));

  // デフォルトのコードページはShift-JIS
  int CodePage = m_AnsiCodePage;

  // Shift-JIS(CP932)を優先してコードページを判別する
  HRESULT result = m_pMLang->DetectInputCodepage(MLDETECTCP_NONE, m_AnsiCodePage,
    pBuffer, &BufferSize, EncodingInfo, &Encodings);
  if (FAILED(result)) return m_AnsiCodePage;

  // 候補の中に識別対象となるコードページがあるか検索する
  for (int i = 0; i < Encodings; ++i) {
    if (m_DetectCP.find(EncodingInfo[i].nCodePage) != m_DetectCP.end()) {
      CodePage = EncodingInfo[i].nCodePage;
      break;
    }
  }
  return CodePage;
}
```

候補となるコードページを最大10個返す

「候補」から有効なコードページを検索

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

std::setの使い方

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- std::setとは、値がキーを兼ねるオブジェクトの集合
  - 重複値を許さない
  - 検索が速い(赤黒木で実装)
  - コンテナは常にソート済み
  - ランダムアクセスは出来ない
  - 自クラスで使用する場合は、operator<()が実装されている必要がある

```
class CTkyParam
{
public:
  CTkyParam(long mc = 0);
  CTkyParam(long mc, double b, double l);
  virtual ~CTkyParam() {}

  long MeshCode; // 3次メッシュコード
  double B, L; // 緯度／経度

  // キーの比較
  bool operator<(const CTkyParam& Param) const {
    return this->MeshCode < Param.MeshCode;
  }
};
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

std::setの使い方

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- コンテナの操作
  - 値を追加するには、insertメソッドを使用する

```
while (infile.eof() == false) {
  infile >> MeshCode >> B >> L;
  TkyParams.insert(CTkyParam(MeshCode, B, L));
}
```

  - データの検索をするには、findメソッドを使用する
    - 値が存在しない場合は末尾を指すイテレータを返す

```
if (TkyParams.find(CTkyParam(46303595)) != TkyParams.end()) {
  std::cout << "MeshID 46303595 is found!" << std::endl;
}
```

  - その他
    - データの削除：eraseメソッド、clearメソッド
    - 要素数の取得：countメソッド

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1時間で作る(?)簡易テキストファイル検索ツール
Step3：正規表現マッチング

VCLと標準C++の文字列クラスの違い

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 標準C++の文字列クラス
  - std::string → AnsiStringに相当
  - std::wstring → UnicodeStringに相当
    - 注意：標準C++の文字列クラスはコードページを持たない std::stringはUTF-8やShift-JISの場合がある。
- VCLの文字列クラスと値のやりとりを行う場合は、必ずオーバーヘッドが発生する
  - std::wstringからUnicodeStringへの代入
    - c_str()関数で取得したポインタを渡す
  - UnicodeStringからstd::wstringへの代入
    - w_str()関数で取得したポインタを渡す

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

VCLと標準C++の文字列クラスの違い

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

コード例：

```
//---------------------------------------------------------------------------
#include <iostream>
#include <vcl.h>
#include <string>
#pragma hdrstop
#include <tchar.h>
//---------------------------------------------------------------------------

#pragma argsused
int _tmain(int argc, _TCHAR* argv[])
{

  std::wstring wstr(L"あいうえお");
  UnicodeString us;

  us = wstr.c_str(); // wstringからUnicodeStringへの代入
  std::cout << us.t_str() << std::endl;

  us = UnicodeString(L"かきくけこ");
  wstr = us.w_str(); // UnicodeStringからwstringへの代入
  std::cout << wstr << std::endl;

  return 0;
}
//---------------------------------------------------------------------------
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

C++Builderで使える主な正規表現ライブラリ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- C++Builderにバンドルされているもの
  - boost::regex
    - ワイド文字列にも対応、C++0xで標準ライブラリに取り込まれる
  - boost::xpressive
    - Regression testの結果がfail
  - TRegexp
    - Undocumented
  - PCRE (Perl Compatible Regular Expressions)
    - ワイド文字列非対応
- 外部
  - 鬼車 (http://www.geocities.jp/kosako3/oniguruma/index_ja.html)
    - Rubyの正規表現エンジンとして採用されている

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::regexを使った正規表現マッチング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- boost::regexの使い方
  1. boost/regex.hppをインクルード
  2. basic_regex(regex/wregex)クラスで正規表現のコンパイルを行う
     - regexクラスはchar型がベース
     - wregexクラスはwchar_t型がベース

```
boost::wregex reg(edtRegex->Text.w_str(), boost::regex::normal);
```

正規表現文字列　　オプション(省略可)

- 主なオプション ：フラグはor(|)で結ぶ

icase	大文字小文字を区別しない
normal	Perlの正規表現を使用
basic	POSIX基本の正規表現を使用
extended	POSIX拡張の正規表現を使用
no_except	不正な式が見つかった場合に例外を投げない

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::regexを使った正規表現マッチング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

3. 検索を行う

検索を行う場合は、regex_search関数かregex_match関数を使用する

```
template<class charT, class Allocator, class traits>
bool regex_search(const charT* str,
                  match_results<const charT*, Allocator>& m,
                  const basic_regex<charT, traits>& e,
                  match_flag_type flags = match_default);
```

str : 文字列

m: マッチ結果

cmatch	文字列がconst char*型
wcmatch	文字列がconst wchar_t*型
smatch	文字列がstd::string型
wsmatch	文字列がstd::wstring型

e : 正規表現

flags : 検索オプション

regex_search関数 : 文字列の一部分がマッチすればtrue

regex_match関数 : 文字列の全てがマッチすればtrue

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::regexを使った正規表現マッチング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

コード例 :

```
// 正規表現のコンパイル
boost::wregex reg(edtRegex->Text.w_str(), boost::regex::normal);

boost::wcmatch m;
boost::wcmatch::iterator iter;

// 文字列が正規表現にマッチするかどうか
if (boost::regex_search(edtString->Text.w_str(), m, reg)) {

  // マッチした部分文字列を出力
  for (iter = m.begin(); iter != m.end(); ++iter) {
    lstSearch->Items->Add(iter->str().c_str());
  }

  lblSearch->Caption = _T("マッチしました。");
} else {
  lblSearch->Caption = _T("マッチしませんでした。");
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::regexを使った正規表現マッチング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

4. 置換を行う

置換を行う場合は、regex_replace関数を使用する

```
template <class traits, class charT>
basic_string<charT> regex_replace(const basic_string<charT>& s,
                                  const basic_regex<charT, traits>& e,
                                  const basic_string<charT>& fmt,
                                  match_flag_type flags = match_default);
```

s : 文字列
e : 正規表現
fmt : 書式化文字列
flags : 置換時のオプション
　format_first_only:　最初にマッチした場所のみ置換
　format_literal:　文字列はリテラル

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::regexを使った正規表現マッチング

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

コード例 :

```
void __fastcall TfrmReplace::btnReplaceClick(TObject *Sender)
{
  // 正規表現のコンパイル
  boost::wregex reg(edtRegex->Text.w_str(), boost::regex::normal);

  std::wstring s(edtString->Text.w_str()); // 対象文字列
  std::wstring format(edtFormat->Text.w_str()); // 書式化文字列

  boost::match_flag_type f = boost::match_default;
  if (chkReplaceFirst->Checked) {
    // 最初のマッチ部分のみ置換
    f |= boost::format_first_only;
  }
  if (chkLiteral->Checked) {
    // 文字列はリテラル
    f |= boost::regex_constants::format_literal;
  }

  // 置換の実行
  std::wstring result = boost::regex_replace(s, reg, format, f);

  edtResult->Text = result.c_str();
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1時間で作る(?)簡易テキストファイル検索ツール

Step4 :検索結果の表示

データ構造

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 検索結果保持に必要なデータ
  - ファイル名
  - 行番号
  - コードページ
- これらをboost::tupleとstd::vectorを使用して結果を保存する

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::tupleとは

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 複数の値をセットとして扱うコンテナ
- 構造体のように定義を必要としない
  - ただし、定義が長くなるのでtypedefで別名をつけると便利

```
#include <boost/tuple/tuple.hpp>

typedef boost::tuple<UnicodeString, int, UnicodeString> GrepRec;
```

UnicodeString, int, UnicodeStringの組を１つの変数として扱う

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

boost::tupleとは

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 生成

```
// 文字列、整数、文字列からなるタプル
boost::tuple<UnicodeString, int, UnicodeString> rec(FileNameFull, i + 1, Encoding);
```

コンストラクタに値を並べて渡すだけ

- アクセス
  - 各要素へのアクセスはget<n>()メソッドを使用

```
boost::tuple<UnicodeString, int, UnicodeString> Rec;
UnicodeString FileName = rec.get<0>();      // 0番目の要素(UnicodeString)
int FileLine = rec.get<1>();                // 1番目の要素(int)
UnicodeString Encoding = rec.get<2>();      // 2番目の要素(UnicodeString)
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

ファイル検索部分の実装

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
void __fastcall TfrmMain::ExecGrep(const UnicodeString& BasePath, const UnicodeString& FileName,
    const boost::wregex& reg)
{
  std::unique_ptr<TStringStream> pStream(new TStringStream());
  pStream->LoadFromFile(BasePath + FileName);

  static CDetectCodepage Detecter;

  //文字コードを判別する
  int BufferSize = std::min(2048i64, pStream->Size);
  int CodePage = Detecter.DetectCodepage(&pStream->Bytes[0], BufferSize);

  // エンコーディングを指定してファイルを読み込む
  TEncoding* pEncoding = TEncoding::GetEncoding(CodePage);
  std::unique_ptr<TStringList> pList(new TStringList());
  if (CodePage == 1200) {
    // UTF-16
    pList->LoadFromStream(pStream.get(), TEncoding::Unicode);
  } else if (CodePage == 1201) {
    // UTF-16(BE)
    pList->LoadFromStream(pStream.get(), TEncoding::BigEndianUnicode);
  } else {
    // エンコーディングがCP51932だったら、CP20932で代替
    if (CodePage == 51932) CodePage = 20932;
    TEncoding* pEncoding = TEncoding::GetEncoding(CodePage);
    pList->LoadFromStream(pStream.get(), pEncoding);
    // TEncoding::GetEncodingで取得したエンコード情報は
    // delete で解放する必要がある
    delete pEncoding;
  }
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

ファイル検索部分の実装（続き）

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
  bool Checked = chkNonMatching->Checked;

  UnicodeString FileNameFull = BasePath + FileName;
     UnicodeString Encoding = Detecter.GetCodePageName(CodePage);

     for (int i = 0; i < pList->Count; ++i) {
       UnicodeString Line = (*pList)[i];

       // 正規表現のマッチング
       if (boost::regex_search(Line.t_str(), reg) != Checked) {

         // 結果を保存
         GrepRec rec(FileNameFull, i + 1, Encoding);
         m_Result.push_back(rec);

         // ListViewにも保存
         TListItem* pItem = lvwFiles->Items->Add();
         pItem->Caption = FileName;
         pItem->SubItems->Add(UnicodeString(rec.get<1>()));
         pItem->SubItems->Add(UnicodeString(rec.get<2>()));
         pItem->SubItems->Add(Line);
       }
     }
   }
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

エディタ起動部分の実装

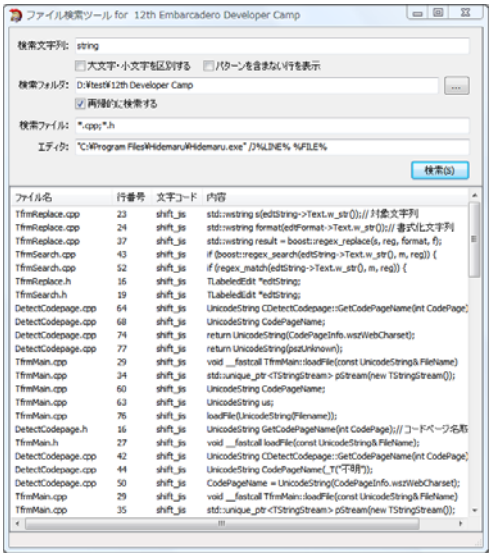
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
void __fastcall TfrmMain::lvwFilesDblClick(TObject *Sender)
{
  int Line = lvwFiles->ItemIndex;
  UnicodeString Command = edtViewerApp->Text;

  // ファイル名・行番号の置換
  UnicodeString FileName = m_Result[Line].get<0>();
  int FileLine = m_Result[Line].get<1>();

  Command = StringReplace(Command, _T("%FILE%"), AnsiQuotedStr(FileName, '"'),
     TReplaceFlags() << rfIgnoreCase);
  Command = StringReplace(Command, _T("%LINE%"), UnicodeString(FileLine),
     TReplaceFlags() << rfIgnoreCase);

  // エディタの起動
  STARTUPINFO sinfo;
  GetStartupInfo(&sinfo);

  PROCESS_INFORMATION pinfo;
  ZeroMemory(&pinfo , sizeof(PROCESS_INFORMATION));
  CreateProcess(NULL, Command.t_str(), NULL, NULL, FALSE,
    CREATE_NEW_PROCESS_GROUP | CREATE_DEFAULT_ERROR_MODE, NULL, NULL, &sinfo, &pinfo);
  CloseHandle(pinfo.hProcess);
  CloseHandle(pinfo.hThread);
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

ひとまず完成！

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

まとめ

まとめ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- STLの積極的使用
  - 一般的なロジックはSTLで実装済み
- 文字コード、エンコーディング
  - エンコーディングの自動認識における諸問題
- C++0xへの移行準備
  - auto_ptrは原則としてunique_ptrに置き換える
  - いくつかのライブラリはBoostから標準C++ライブラリに取り込まれる
    - regex
    - tuple
    - shared_ptr, weak_ptr
    - など

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

最後に・・・

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

ご静聴ありがとうございました

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

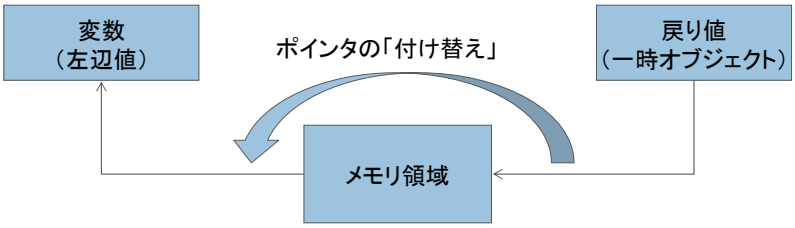
付録

ムーブセマンティクスと右辺値参照について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ムーブセマンティクス（値の付け替え）
  - オブジェクト同士で値をやりとりするときに、代入（メモリ領域のコピー）ではなく、領域のアドレスを付け替えることによって、高速化を図る手法

```
vector<string> foo();

vector<string> StrBuffer = foo();
```



  - STLはムーブセマンティクスを使用した実装で書き換えられる

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

ムーブセマンティクスと右辺値参照について

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 右辺値参照
  - C++0xで導入された新機能の一つ
  - 右辺値とは、関数の戻り値のように変数名と結びついていない値
  - 右辺値を直接参照することによって、無駄なコピーを省くことが出来る

```
// 戻り値（一時オブジェクト）の参照
std::cout << "CFoo&& rfoo = FooFunc();" << std::endl;
CFoo&& rfoo = FooFunc();
std::cout << "rfoo.m_Value = " << rfoo.m_Value << std::endl;
```

- ムーブコンストラクタ
  - 引数に右辺値が渡された場合のコンストラクタ

```
CFoo(CFoo&& foo) :
   m_Value(foo.m_Value)
{
    std::cout << “Move constructor” << std::endl;
    foo.m_Value = 0; // 右辺値が持つリソースの破棄
}
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Boost C++ Librariesのインストール

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1. Boostのアーカイブとboost jamをダウンロード
   http://www.boost.org/users/download/
   boost jamのWindowsバイナリは、boost-jam-3.1.17-1-ntx86.zip
2. Boostのアーカイブを展開
3. boost jamのアーカイブにある、bjam.exeをパスが通ったフォルダにコピー
4. RAD Studio Command Promptを開き、Boostのアーカイブを展開したフォルダで以下を実行

```
bjam --prefix=C:¥lib¥boost¥ --toolset=borland-6.1.0 link=static,shared
runtime-link=static,shared release debug install
```

   注)--prefixの値はインストール先

5. プロジェクトオプションのインクルードパスとライブラリパスを変更すれば、C++Builder標準のBoostと共存可能

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。