MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Luan Gaspar Pinto de Melo

Uso de linguagem de script para realizar operações de álgebra de mapas da TerraLib5

São José dos Campos
2015

Luan Gaspar Pinto de Melo

Uso de linguagem de script para realizar operações de álgebra de mapas da TerraLib5

Monografia da disciplina de Introdução ao Geoprocessamento – (SER300) do Instituto Nacional de Pesquisas Espaciais, ministrada pelo Dr. Antônio Miguel Vieira Monteiro.

Professora Orientadora: Lúbia Vinhas

São José dos Campos
2015

Resumo

Atualmente existem diferentes Sistemas de Informações Geográficas com características próprias. Alguns exemplos destes sistemas são o ArcGis, o QuantumGis, o Spring e o TerraView. Muitos desses sistemas oferecem a possibilidade de acesso aos seus algoritmos e funcionalidades através de de programas escritos em linguagem de mais alto nível, tipicamente linguagens de script. Com a oferta de uma linguagem de programação ao além de interfaces gráficas, é possível se obter mais flexibilidade e eficiência na implementação de metodologias que combinam diferentes dados para obtenção de um resultado específico, por exemplo através de operações de álgebra de mapas. A proposta desse trabalho será estudar as funcionalidades de álgebras de mapas existentes na biblioteca TerraLib5 e propor uma API de acesso a elas em uma linguagem de script a ser definida (ex. LUA, Python ou Java). A linguagem LEGAL existente do SPRING será usada como referência. Alguns estudos de uso serão feitas e será produzida uma comparação entre os dois ambientes.

Palavras chave: Álgebra de mapas, SIGs, TerraLib5, Linguagem de Script.

Sumário

1. Introdução

1.1 Sistema de Informação Geográfica (SIG)

A coleta de informações sobre a distribuição geográfica de recursos minerais, propriedades, animais e plantas sempre foi uma parte importante das atividades das sociedades organizadas. Antigamente, no entanto, isto era feito apenas em documentos e mapas em papel; isto impedia uma análise que combinasse diversos mapas e dados. Com o desenvolvimento simultâneo, da tecnologia de Informática, tornou-se possível armazenar e representar tais informações em ambiente computacional, abrindo espaço para o aparecimento de Geoprocessamento (Câmara e Davis, 2015).

Nesse contexto, o termo Geoprocessamento denota a disciplina do conhecimento que utiliza técnicas matemáticas e computacionais para o tratamento da informação geográfica que influenciou as áreas de Cartografia, Análise de Recursos Naturais, Transportes, Comunicações, Energia e Planejamento Urbano e Regional. As ferramentas computacionais para Geoprocessamento, chamadas de Sistemas de Informação Geográfica (SIG), permitem realizar análises complexas, ao integrar dados de diversas fontes e ao criar bancos de dados georreferenciados. Tornando-se possível automatizar a produção de documentos cartográficos (Câmara e Davis, 2015).

Neste trabalho são usados como exemplos dois sistemas de informação geográfica: Spring e TerraView. Ambos são implementados com linguagem de programação C++. C++ é uma evolução da linguagem C e é considerada uma linguagem híbrida, pois suporta a programação imperativa e orientada a objeto.

1.2 Álgebras de Mapas

O que diferencia um Sistema de Informação Geográfica de outros tipos de sistemas de informação são as funções que realizam análises espaciais. Tais funções utilizam os atributos espaciais e não espaciais das entidades gráficas armazenadas na base de dados espaciais e buscam fazer modelos sobre os fenômenos do mundo real, seus aspectos ou parâmetros (Cordeiro et al., 2015)

As diversas operações de análise geográfica são divididas em: operadores sobre objetos, operadores sobre campos, operadores de transformação

entre campos e objetos e operadores mistos entre objetos e campos. O tema “álgebra de mapas” foi popularizado a partir dos livros “Geographic Information System and Cartographic Modeling” de Tomlin (Cordeiro et al., 2015).

Os elementos da álgebra de mapas descrita por Tomlin consistem de mapas que associam a cada local de uma dada área de estudo um valor quantitativo (escalar, ordinal, cardinal ou intercalar) ou qualitativo (nominal) (Cordeiro et al., 2015).

Uma das operações mais simples é o “BUFFER”. Essa operação é utilizada para gerar um polígono com uma determinada distância a partir de um objeto. Esta operação tem apenas um operando e poucos parâmetros de entrada.

1.3 TerraLib5

TerraLib é uma biblioteca C++ base de software para desenvolver Sistemas de Informação Geográfica (SIG). É desenvolvido pelo Instituto Nacional de Pesquisas Espaciais (INPE), Brasil. TerraLib é software livre e open source. Você pode redistribuí-lo e / ou modificá-lo sob os termos da GNU Lesser General Public License como publicado pela Free Software Foundation (INPE, 2015a).

1.4 Linguagem de script Python

O Python é uma linguagem de programação de propósito geral, frequentemente aplicada em funções de script. Ela é comumente definida como uma linguagem de script orientada a objetos – uma definição que combina suporte para POO com orientação global voltada para funções de script. Na verdade, as pessoas frequentemente usam o temo “script”, em vez de “programa”, para descrever um arquivo de código Python. “script” descreve um arquivo de nível superior mais simples e “programa” se refere a um aplicativo de vários arquivos mais sofisticado (Lutz e Ascher, 2004).

Como o termo “script” tem significados diferentes para diferentes observadores, alguns preferem que ele não ele não seja aplicado ao Python. Na verdade, as pessoas tendem a pensar em três definições muito diferentes quando ouvem o Python rotulado como uma linguagem de “script”, algumas das quais são mais úteis do que outras: Ferramentas de shell, Linguagem de controle e Facilidade de uso. Em geral, o termo script provavelmente é melhor usado para descrever o modo de desenvolvimento rápido e flexível que o Python suporta (Lutz e Ascher, 2004).

2. Objetivo

Este trabalho visa estudar como álgebras de mapas podem se apresentar para o usuário final e discutir quais as implicações das diferentes maneiras existentes. Com isso é proposta uma API Python que obtenha acesso à álgebra de mapas utilizando as funcionalidades fornecidas pela TerraLib5. O objetivo desta proposta é possibilitar uma maior flexibilidade e eficiência na implementação de metodologias que combinam diferentes dados para obtenção de um resultado específico. Por exemplo, realizar várias operações de álgebra de mapas com um único script.

Isso pode tornar mais fácil a utilização para alguém que não tem o domínio da computação. Ou seja, permitir que geógrafos, biólogos e outros profissionais, possam ter acesso à funcionalidades de álgebra de mapas em uma linguagem de alto nível, que pode ser aprendida mais facilmente

3. Metodologia

Para este trabalho será utilizada a operação BUFFER por ser uma das operações de álgebra de mapas mais simples. Uma exemplificação é adotada para o rio Paraíba do Sul na região do município de Volta Redonda - RJ, que é mestrado na Figura 1.

Figura 1. Região do município de Volta Redonda – RJ (Google Earth, 2015).

O polígono do rio Paraíba do Sul nesta região é apresentado na Figura 2.

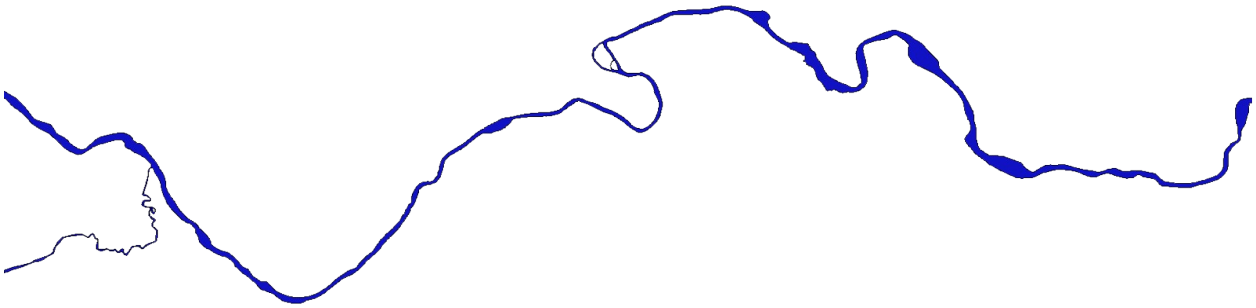


Figura 2. Polígono do rio Paraíba do Sul (PortalVR, 2015)

Para obter o BUFFER do polígono mostrado na Figura 2 é possível utilizar *softwares* como o Spring, TerraView, ArcGis e Quantum Gis. Estes *softwares* fornecem uma interface gráfica para que o usuário possa realizar operações de álgebras de mapas. Neste trabalho é mostrado um exemplo da operação buffer utilizando a interface do Spring e do TerraView.

3.1 Spring

O resultado do BUFFER obtido através da ferramenta "mapa de distância" do Spring é mostrado na Figura 3.

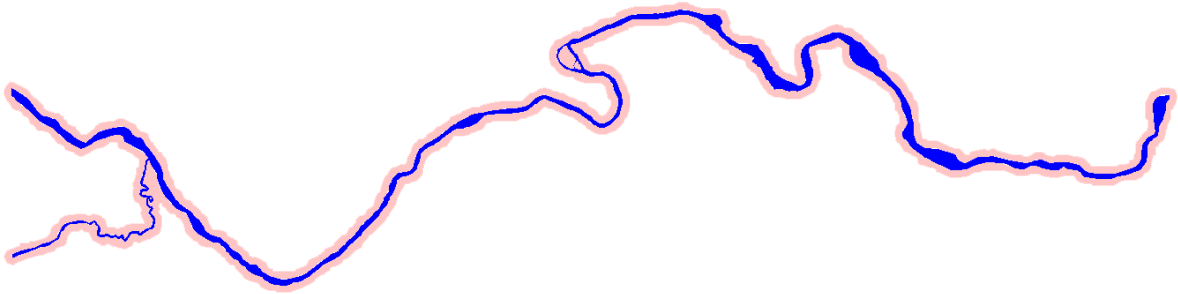


Figura 3. Resultado do Mapa de distância no Spring

3.2 TerraView

O resultado do BUFFER obtido através da ferramenta "Operações vetoriais" do TerraView é mostrado na Figura 4.

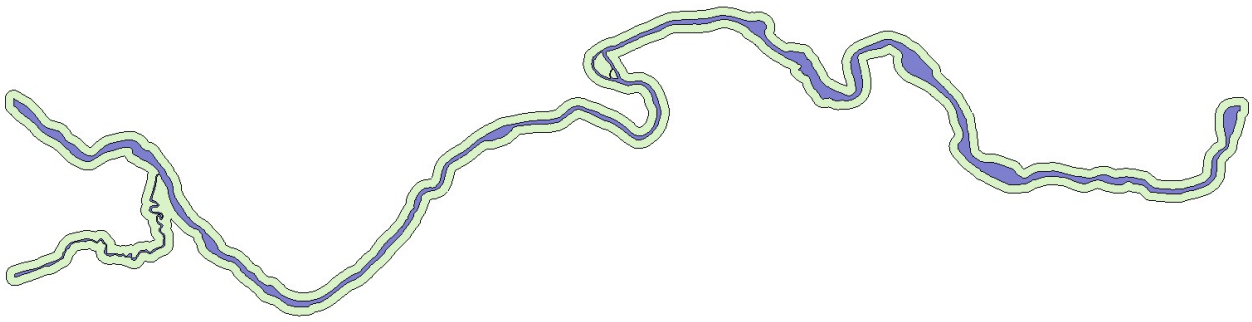
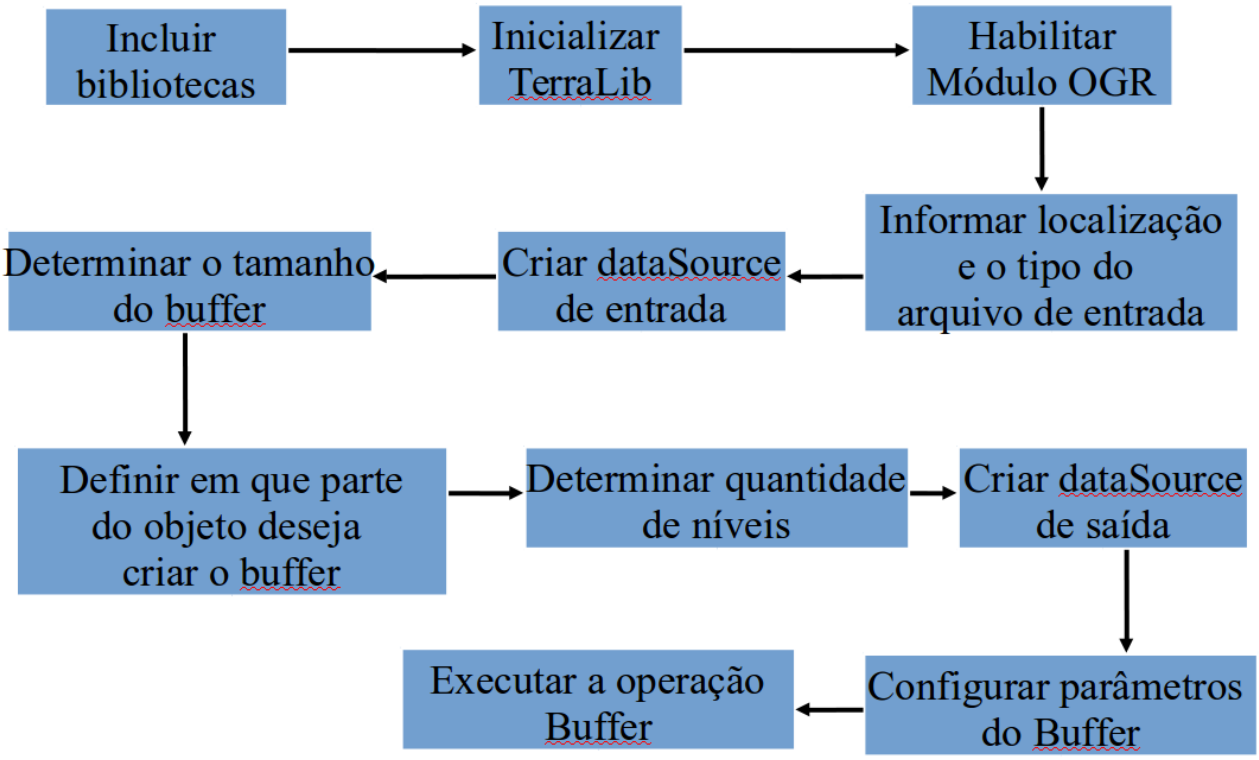


Figura 4. Resultado da operação BUFFER no TerraView

3.3 C++

O que acontece quando o usuário clica em uma botão para gerar o BUFFER? Dentro desses sistemas citados é desencadeado um conjunto de ações que está pré programado em C++. um exemplo de um trecho do código executado pelo TerraView é apresentado no Apêndice A. De acordo com a Figura 5, este trecho de código importa e ativa a biblioteca Terralib, inicializa o módulo OGR que é responsável pela decodificação de dados vetoriais e a localização e o tipo de arquivo de entrada é informado. Após isso, um dataSource de entrada é criado e são determinados a distância do buffer, a parte do objeto que o buffer será criado e a quantidade de níveis. Além disso, um dataSource de saída é criado e os parâmetros do buffer são configurados. Se os parâmetros são válidos, então a operação buffer é executada.Figura 5. Fluxograma do código para realiar a operação BUFFER.



A Figura 6 mostra o resultado do Buffer obtido através do código C++.

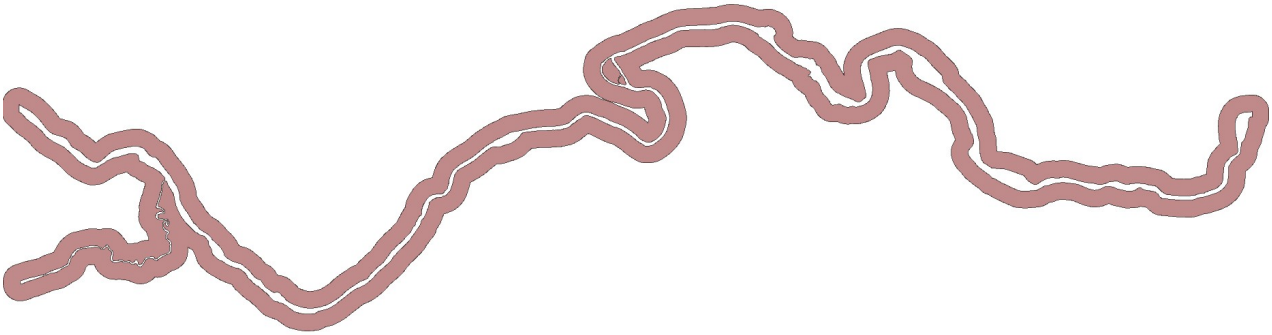


Figura 6. Buffer obtido através do programa em C++.

3.4 Script

Quando é necessário realizar diversas operações, a interface gráfica exige que o usuário faça cada operação separadamente. Isso se torna muito repetitivo, desgastante e inviável. Por esse motivo, a maioria dos SIGs oferecem uma linguagem de script para que o usuário consiga encadear uma sequência de operações de forma mais direta e automática.

A escolha de usar uma linguagem de script, ao invés de usar a própria linguagem C++ para encadear uma sequência de operações, é devido a facilidade e simplicidade que este tipo de linguagem oferece.

4. Desenvolvimento

Primeiramente, é necessário configurar e aprender sobre a TerraLib. Para isso, instalou-se todas as bibliotecas de terceiros que são dependências da TerraLib, tais como, Boost, Git, QT e QWT. Baixou-se a TerraLib do repositório GIT (INPE, 2015b) e compilou-se na plataforma Linux Ubuntu 14.04.02 LTS. Configurou-se a IDE Eclipse para fazer o link com a TerraLib e adicionar os módulos utilizados na aplicação. Também configurou-se no Eclipse as variáveis de ambiente e desenvolveu-se o código para realizar a operação Buffer. Esse código é apresentado no Apêndice B.

Para utilizar a linguagem de script Python para a operação Buffer, é necessário analisar os parâmetros de entrada e adaptar o código C++. Posteriormente, é preciso gerar um Bind, que é o módulo gerado em C++ e compilado e instalado na biblioteca do Python.

Um exemplo de como seria um script em Python para executar a operação Buffer é apresentado na Figura 7.

```
Import buffer

BufferOGRToOGR(arquivo, posicaoBuffer, regraLimite, niveis, arquivoSaida)
```

Figura 7. Exemplo de script no Python para gerar um Buffer.

Na Figura 7, arquivo corresponde ao caminho completo do arquivo de

entrada, posicaoBuffer é a parte do objeto onde será criado o Buffer (interno, externo ou ambos), regraLimite representa o tamanho do Buffer a ser criado, niveis define a quantidade de camadas de Buffer e arquivoSaida corresponde ao nome do arquivo de saída.

5. Resultados e conclusões

Com os estudos realizados, o autor deste trabalho compreendeu a linguagem de programação C++, que é considerada uma das linguagens mais difíceis de se aprender. Adicionalmente, o autor absorveu os conceitos da biblioteca TerraLib e conseguiu programar utilizando essa biblioteca. Portando, o autor compreendeu o domínio da aplicação.

Este é um trabalho inicial que possivelmente será tema da proposta de dissertação do autor. Para trabalhos futuros pretende-se criar os Binds para Python e possibilitar o uso desta linguagem para realização de operações de álgebras de mapas.

Referências

Câmara, G.; Davis, C. **Introdução.** <http://www.dpi.inpe.br/gilberto/livro/introd/cap1-introducao.pdf>. Acesso em: 06 mai. 2015.

Cordeiro, J.P.; Barbosa, C.C.F.; Câmara, G. **Álgebra de campos e objetos.** <http://www.dpi.inpe.br/gilberto/livro/introd/cap8-algebra.pdf>. Acesso em: 06 mai. 2015.

Google Earth. <https://www.google.com.br/maps/place/Volta+Redonda+-+RJ/@-22.5052939,-44.0893257,17370m/data=!3m1!1e3!4m2!3m1!1s0x9ea2ac4b4e5c1d:0xd475bc9356fcad22!6m1!1e1?hl=pt-BR>. Acesso em: 08 jun. 2015.

INPE. **TerraLib and TerraView 5.0 Wiki Page.** <http://www.dpi.inpe.br/terralib5/wiki/doku.php>. Acesso em: 10 jun. 2015a.

INPE. **TerraLib 5.0 and TerraView 5.0 - Download Source Code**. <http://www.dpi.inpe.br/terralib5/wiki/doku.php?id=wiki:terralib50_download>. Acesso em: 30 abr. 2015b.

Lutz, Mark; Ascher, David. **Aprendendo Python.** São Paulo:Artmed, 2004.

PortalVR. **Arquivo shapefile**. <http://www.portalvr.com/geoprocessamento/mod/mapas-georeferenciados/shapefile/>. Acesso em: 20 mai. 2015.

Apêndices

Apêndice A: trecho do código executado pelo TerraView

```
	std::string inDsetName = "rio_paraiba_sul";


	if (!srcDs->dataSetExists(inDsetName)) {
		std::cout << "Input dataset not found: " << inDsetName << std::endl;
		return false;
	}
	std::auto_ptr<te::da::DataSet> inDset = srcDs->getDataSet(inDsetName);
	std::auto_ptr<te::da::DataSetType> inDsetType = srcDs-
>getDataSetType(inDsetName);
	double distance = 0.004;
	int bufferPolygonRule = te::vp::ONLY_OUTSIDE;
	int bufferBoundariesRule = te::vp::DISSOLVE;
	bool copyInputColumns = false;
	int levels = 1;

	std::string file_result = data_dir+
"/file_result_rio_paraiba_sul_cplusplus.shp";
	std::map<std::string, std::string> tgrInfo;
	tgrInfo["URI"] = file_result;
	tgrInfo["DRIVER"] = "ESRI Shapefile";

	te::da::DataSourcePtr trgDs(te::da::DataSourceFactory::make("OGR"));
	trgDs->setConnectionInfo(tgrInfo);
	trgDs->open();

	std::string outDS = "file_result_rio_paraiba_sul_cplusplus";
	if (trgDs->dataSetExists(outDS)) {
		std::cout << "A dataset with the same requested output dataset name
already exists: " << outDS << std::endl;
		return false;
	}

	te::vp::BufferOp* bufferOp = new te::vp::BufferMemory();
	bufferOp->setInput(srcDs, inDsetName, inDsetType);
	bufferOp->setOutput(trgDs, outDS);
	bufferOp->setParams(distance, bufferPolygonRule, bufferBoundariesRule,
copyInputColumns, levels);

	bool result;
	if (!bufferOp->paramsAreValid())
		result = false;
	else
		result = bufferOp->run();
	delete bufferOp;
	return result;
```

Apêndice B: Código em C++ para realizar a operação Buffer

```
#include <terralib/common.h>
#include <terralib/dataaccess.h>
#include <terralib/dataaccess/datasource/DataSourceFactory.h>
#include <terralib/vp/BufferMemory.h>
#include <terralib/vp/BufferOp.h>
#include <terralib/vp/BufferQuery.h>
#include <iostream>
#include <map>
#include <memory>
#include <string>
#include <vector>

bool BufferOGRToOGR() {
	std::string data_dir =
"/home/luan/Documentos/Geoprocessamento/rio_paraiba_sul";
	std::string filename = data_dir + "/rio_paraiba_sul.shp";

	std::map<std::string, std::string> srcInfo;
	srcInfo["URI"] = filename;
	srcInfo["DRIVER"] = "ESRI Shapefile";

	te::da::DataSourcePtr srcDs(te::da::DataSourceFactory::make("OGR"));
	srcDs->setConnectionInfo(srcInfo);
	srcDs->open();

	std::string inDsetName = "rio_paraiba_sul";
	if (!srcDs->dataSetExists(inDsetName)) {
		std::cout << "Input dataset not found: " << inDsetName << std::endl;
		return false;
	}
	std::auto_ptr<te::da::DataSet> inDset = srcDs->getDataSet(inDsetName);
	std::auto_ptr<te::da::DataSetType> inDsetType = srcDs-
>getDataSetType(inDsetName);
	double distance = 0.004;
	int bufferPolygonRule = te::vp::ONLY_OUTSIDE;
	int bufferBoundariesRule = te::vp::DISSOLVE;
	bool copyInputColumns = false;
	int levels = 1;

	std::string file_result = data_dir+
"/file_result_rio_paraiba_sul_cplusplus.shp";
	std::map<std::string, std::string> tgrInfo;
	tgrInfo["URI"] = file_result;
	tgrInfo["DRIVER"] = "ESRI Shapefile";

	te::da::DataSourcePtr trgDs(te::da::DataSourceFactory::make("OGR"));
	trgDs->setConnectionInfo(tgrInfo);
	trgDs->open();

	std::string outDS = "file_result_rio_paraiba_sul_cplusplus";
	if (trgDs->dataSetExists(outDS)) {
		std::cout << "A dataset with the same requested output dataset name
already exists: " << outDS << std::endl;
		return false;
	}

	te::vp::BufferOp* bufferOp = new te::vp::BufferMemory();
```

```
      bufferOp->setInput(srcDs, inDsetName, inDsetType);
      bufferOp->setOutput(trgDs, outDS);
      bufferOp->setParams(distance, bufferPolygonRule, bufferBoundariesRule,
copyInputColumns, levels);

      bool result;
      if (!bufferOp->paramsAreValid())
            result = false;
      else
            result = bufferOp->run();
      delete bufferOp;
      return result;
}



#include <terralib/common/PlatformUtils.h>
#include <terralib/common.h>
#include <terralib/plugin.h>
#include "VectorProcessing.h"
#include <cassert>
#include <cstdlib>
#include <exception>
#include <iostream>



int main(int argc, char** argv)
{
  try
  {
    TerraLib::getInstance().initialize();

    te::plugin::PluginInfo* info;

    std::string plugins_path =
te::common::FindInTerraLibPath("share/terralib/plugins");

#ifdef TERRALIB_MOD_OGR_ENABLED
    info = te::plugin::GetInstalledPlugin(plugins_path + "/te.da.ogr.teplg");
    te::plugin::PluginManager::getInstance().add(info);
#endif

    te::plugin::PluginManager::getInstance().loadAll();

        std::cout << std::endl << "Buffer OGR to OGR: " << std::endl;
    if (BufferOGRToOGR())
      std::cout << "\tOK!" << std::endl;

    te::plugin::PluginManager::getInstance().unloadAll();

    TerraLib::getInstance().finalize();
  }
  catch(const std::exception& e)
  {
    std::cout << std::endl << "An exception has occurred: " << e.what() <<
std::endl;

    return EXIT_FAILURE;
```

```
  }
  catch(...)
  {
    std::cout << std::endl << "An unexpected exception has occurred!" <<
std::endl;

    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```