**Relatório de Linguagens de Programação**

**Trabalho 2**

Nomes: Douglas Dallavale e Vinícius Parmeggiani

**Introdução**

O presente relatório tem por objetivo mostrar os diferentes casos testes que serão utilizados na próxima etapa do trabalho, bem como seus respectivos resultados em C++.

**Testes**

**Teste 1**

**Versão de Entrada**

datatype String

package my.company.common {

entity HasAuthor {

author: String

}

}

package my.company.blog {

import my.company.common.\*

entity Blog {

title: String

many posts: Post

}

entity Post extends my.company.common.HasAuthor {

title: String

content: String

many comments: Comment

}

entity Comment extends HasAuthor {

content: String

}

}

**Versão de saída em C++**

#include<common>

#include <vector>

using namespace blog;

class HasAuthor{

std::string m\_Author;

public:

HasAuthor(std::string author) : m\_Author(author){};

std::string getAuthor(){ return m\_Author; }

}

class Blog

{

std::string m\_Title;

std::vector<Post> m\_Posts;

public:

Blog(std::string title,

std::vector<Post> posts) :

m\_Title(title),

m\_Posts(posts){};

std::string getTitle(){ return m\_Title; }

std::vector<Post> getPosts(){ return m\_Posts; }

}

class Post : HasAuthor

{

std::string m\_Title;

std::string m\_Content;

std::vector<Comment> m\_Comments;

public:

Post(std::string title,

std::string content,

std::vector<Comment> comments) :

m\_Title(title),

m\_Content(content),

m\_Comments(comments){};

std::string getTitle(){ return m\_Title; }

std::string getContent(){ return m\_Content; }

std::vector<Comment> getComments(){ return m\_Comments; }

}

class Comment : HasAuthor

{

std::string m\_Content;

public:

Comment(std::string content) : m\_Content(content){};

}

**Teste 2**

**Versão de Entrada**

datatype Int

entity Number {

number: Int

}

entity Func extends Number {

many numbers: Number

}

**Versão de saída em C++**

#include <vector>

class Number {

int m\_Number;

public:

Number(int number) : m\_Number(number){};

int getNumber(){ return m\_Number; }

}

class Func : Number{

std::vector<int> numbers;

public:

Func(std::vector<int> numbers) : m\_Numbers(numbers){};

std::vector<int> getNumbers(){ return m\_Number;}

}

**Teste 3**

**Versão de Entrada**

datatype String

entity Names{

many names: String

}

**Versão de saída em C++**

#include <vector>

class Names{

vector<std::string> m\_Names;

public:

Names(std::vector<std::string> names) :

m\_Names(names){};

std::string getNames(){ return m\_Names; }

}

**Referencias**

<https://www.eclipse.org/Xtext/documentation/103_domainmodelnextsteps.html>

<https://www.eclipse.org/Xtext/documentation/102_domainmodelwalkthrough.html>