Gem #116: Ada и исключения C++

Автор: Quentin Ochem, AdaCore

Краткое содержание: Одна из главных проблем, с которой можно столкнуться в ходе сопряжения Ada и C++, - это действие исключений одного компилятора в другом. В данном Gem будет продемонстрировано, как новый механизм работы с исключениями, реализованный в GNAT, способствует работе с исключениями одного языка в другом. Следует принять во внимание, что приведённый код будет работать только в версиях GNAT начиная с Pro 7.

Давайте начнём…

Сегодня мы решили повеселиться, поэтому и создали приложение, которое сочетает Ada и C ++ код. Все идёт нормально. Мы использовали генератор связывания C ++ для Ada для привязки классов C ++ непосредственно к Ada и расширили их. Тем не менее, есть что-то, что нужно тщательно рассмотреть - что произойдёт, если исключение будет вызвано / поднято кодом C ++ или Ada? Попробуем:

Шаг 1 - Код C ++

Мы собираемся написать очень простой код на C ++, только два класса, один наследующий от другого и переопределяющий «вычисляемый» примитив:

```
class COperation {
  public:
  virtual int compute (int a, int b);
  COperation ();
};

class CDivision : public COperation {
  public:
  virtual int compute (int a, int b);
  CDivision ();
};

int CDivision::compute (int a, int b) {
  if (b == 0) {
    throw new Problem ("Division by 0 in C++!");
  } else {
    return a / b;
  }
}
```

При вычислении `CDivision` мы создадим исключение C ++, если произойдёт деление на ноль. Давайте добавим вторую подпрограмму динамической диспетчеризации:

```
void cpp_main (COperation & op) {
  try {
     cout << op.compute (1, 0);
  } catch (...) {
    cout << "Unknown exception caught, rethrowing..." << "
";
    throw;
  }
}
```

Эта функция вызывает операцию вычисления, получает исключение, чтобы отобразить сообщение, а затем повторяет его.

Теперь мы свяжем это с Ada. Предполагая, что все спецификации находятся в файле `base.hh`, простой вызов `gcc` сделает это:

```
g++ -fdump-ada-spec base.hh
```

Шаг 2 - Расширение кода на C ++ в Ada

В C ++ реализована реализация `CDivision`. После фазы связывания, давайте реализуем тот же код в Ada:

```
type Ada_Division is new base_hh.Class_COperation.COperation with
     null record;

function Compute
  (this : access Ada_Division;
   a : int;
   b : int) return int is
begin
   if b = 0 then
      raise Constraint_Error with "Division by 0 in Ada!";
   end if;
   return a / b;
end Compute;
```

Теперь у нас есть три класса в приложении. Один корневой класс C ++, один чистый C ++-потомок и один смешанный потомок Ada / C ++. Давайте посмотрим, как все работает.

Шаг 3 - Захват исключения C ++ в Ada

Исключения C++ не имеют известного имени, как только они достигают мира Ada. Они действуют так, как если бы они были объявлены в теле пакета, поэтому единственный способ поймать их - это использовать общий обработчик исключений. Рассмотрим следующий код:

```
   T_Cpp : aliased base_hh.Class_CDivision.CDivision;
   X : Interfaces.C.Int;
begin
   X :=  T_Cpp.compute (1, 0);
exception
   when others =>
      Put_Line ("[1] Exception caught...");
end;
```

Это напечатает «[1] Exception caught ...» на экране.

Шаг 4 - Обработка исключений на разных языках

Теперь давайте будем немного более амбициозными. Мы собираемся вызвать `cpp_main` функцию с объектом, прибывающим от Ada:

```
   T_Ada : aliased Base_Ada.Ada_Division;
begin
   base_hh.cpp_main (T_Ada'Access);
exception
   when Constraint_Error =>
      Put_Line ("[2] Constraint_Error caught...");
   when others =>
      Put_Line ("[2] Exception caught...");
end;
```

Теперь, оглядываясь назад на реализацию cpp_main, мы вызываем compute с (1, 0) в качестве аргументов. Это вызовет исключение из реализации Ada, которое сначала будет перехвачено кодом C++, печатающим на экране ”неизвестное перехваченное исключение...". То же самое исключение затем переосмысливается стороной C++, возвращаясь в Код Ada выше, печатая "[2] Constraint_Error catched...".

Связанный со статьёй текст программы

Attached Files отсутствуют

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Quentin Ochem имеет опыт разработки программного обеспечения, специализирующийся на разработке программного обеспечения для критически важных приложений. Он имеет более чем 10-летний опыт разработки Ada. Сегодня он работает техническим менеджером AdaCore по проектам, связанным с авионикой, железнодорожной, космической и оборонной отраслями. Он также обучает стандарту DO-178B авионики в университете EPITA в Париже.

Last Updated: 10/13/2017
Posted on: 2/1/2012

Обсуждение…

4 ответов на "Gem # 116: Ada и C ++ Exceptions"

1. 18 января 2012 года в 13:40

Christoph Grein сказал:

Хм, я попробовал ваш код и получил следующее. Так как я совсем не знаю, я застрял. Пожалуйста, помогите с синтаксисом C ++.

```
gprbuild -d -PC:\Users\Grein\Documents\Christoph\Ada\C++\gem_116.gpr base.cpp
g++ -c base.cpp
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp: In member function ‘virtual int CDivision::compute(int,
int)’:
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp:5:15: error: expected type-specifier before ‘Problem’
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp:5:15: error: expected ‘;’ before ‘Problem’
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp: In function ‘void cpp_main(COperation&)’:
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp:13:6: error: ‘cout’ was not declared in this scope
C:\Users\Grein\Documents\Christoph\Ada\C++\base.cpp:15:5: error: ‘cout’ was not declared in this scope
gprbuild: *** compilation phase failed

[2012-01-18 12:35:53] process exited with status 4 (elapsed time: 01.16s)
```

2. 18 января 2012 года в 13:59

Quentin Ochem сказал:

Привет, Christoph,

Это немного сложно сказать, не глядя на фактический файл, но обратите внимание, что показанный код извлекается из полного приложения. В частности, я не предоставлял класс «`Problem`», используемый для исключения. Вероятно, это проблема. Определить что-то вроде:

```
class Problem {
public:
Problem (char *) {}
};
```

вероятно, улучшит ситуацию.

3. 18 января 2012 года в 18:47

Christoph Grein сказал:

Quentin,

спасибо за добавление, но я до сих пор не понимаю, как это связано.

Поскольку это, главным образом, сайт Ada, я думаю, что должен быть предоставлен полный ссылочный пример. Я думаю, что есть много других, не владеющих C ++, но которые в конечном итоге ссылаются на предоставленный код C ++.

Christoph

4. 20 января 2012 года в 11:34

Maciej Sobczak сказал:

Quentin,

Я должен сказать, что этот пример не очень убедителен.

Одной из причин этого является то, что часть `C ++` плохо написана. Очень плохая практика бросать указатели; обрабатывать исключения как (полиморфные) значения. Ещё одна небольшая проблема - терминология - класс `C ++` не является «чистым». Класс `C ++` может быть «абстрактным», если он имеет «чистую» функцию, но даже это не так в этом примере - оба класса `C ++` здесь не абстрактны и просто не имеют определений для своих функций.

Но это детали. Реальная проблема заключается в том, что не рекомендуется насильно связывать `C ++` и Ada на этом уровне.

Гораздо более надёжный подход может заключаться в создании промежуточного слоя `C` (в смысле `extern «C»`) и использования его из Ada. Это может потребовать немного более кодирования, особенно если вам нужно сохранить полиморфизм типов и исключений, но с тем, чтобы иметь окончательный контроль над тем, что происходит. В частности, исключение содержит некоторую (возможно, важную) информацию в нем, и потерять её через границу - это не то, что я считаю правильным решением для привязки модуля.

С Уважением,