「The Future of Fortran」の Doctor Fortran

この記事は、インテル® デベロッパー・ゾーンに公開されている「Doctor Fortran in "The Future of Fortran"」の日本語参考訳です。

2014 年 11 月に、私は、SC14 (以前「Supercomputing」と呼ばれていたイベント) で「The Future of Fortran (Fortran の将来)」と題したセッションを行いました。私は、このセッションに Fortran 標準委員会のほかのベンダーやメンバーの代表の 1 人として参加するように依頼を受けたと思っていたのですが、いざセッションが始まると、そこにいた関係者は私 1 人だったのです! まぁ、それはさておき。

私は、Fortran 標準規格の現状と、現在 Fortran 2015 (この名称になることを期待しています) と呼ばれている次の標準規格について、ベンダー・ニュートラルの中立的なプレゼンテーションを準備していました。質疑応答は、予想よりもはるかに本格的でした (時間が短いにもかかわらず質問者は 70～80 名ほどいました)。セッションの最後に出席者の皆さんにお願いした、Fortran の使用法に関するオンライン調査の結果は興味深いものでした (この記事の最後を参照)。

まず最初に、私は、現在の標準規格 Fortran 2008 (2010) と 1 つ前の標準規格 Fortran 2003 (2004) について話しました。Fortran 2008 を完全にサポートしているベンダーは 1 社 (Cray*)、Fortran 2003 をサポートしているベンダーは 3 社 (IBM*、インテル、PGI*/NVidia*) あります。

Fortran 2015 は、2 つの大きな機能セットの追加と不整合部分の修正のみを含む小規模な改訂として考えられていました。いつものように、いくつかの小さな機能が追加されましたが、現時点ではまだ許容範囲です。すべての変更は、新しい標準規格の Introduction にリストされます。現在のドラフト (英語) はこちらからダウンロードできますが、まだすべての変更は反映されていません。

現在のスケジュールでは、「技術的な作業」は 2015 年に完了する予定です。予想では新しい標準規格は 2017 年に承認される見込みですが、この後に述べるように、さらに遅れる可能性があります。

次のセクションでは大きな変更について概要を説明します。詳細は、ドラフト (上記参照) をご覧ください。

「Interop (相互運用) TS」

最初の大きな機能セットは個別の技術仕様書 TS29113、「Further Interoperability with C」で定義されました。この機能は、最終的な標準規格は非互換にならないという保証のもとで、ベンダーが実装できるように 2012 年に承認されました。「Interop TS」が追加された大きな理由は、MPI 3.0 の必要性でした。タイトルでほのめかされているように、Fortran 2015 では、Fortran 2003 で最初に実装された C との相互運用機能が次のように拡張されました。

- 型引き継ぎおよびランク引き継ぎにより、C の「void *」との相互運用がより簡単になりました。
- 新しいデータ構造「C 記述子」を使用して、ALLOCATABLE、POINTER、形状引き継ぎ、CHARACTER(*) 引数を C に渡したり、C から渡すことができます。
- C 記述子と操作用のルーチンは新しい ISO_Fortran_binding.h C ヘッダーファイルで宣言します。
- OPTIONAL 引数が相互運用可能になりました。
- ASYNCHRONOUS が I/O 以外に拡張されました。
- 仮引数の制限が緩和されました。

型引き継ぎ変数は新しい構文 TYPE(*) で宣言され、仮引数としてのみ使用できます。型引き継ぎ変数 (無制限の多相化変数であり、任意の型を渡すことができます) には型は含まれませんが、通常の多相化変数とは異なり、SELECT TYPE を利用することはできません。Fortran コードで利用するには、C_LOC および C_F_POINTER を使用して Fortran ポインター型に「キャスト」する必要があります。型引き継ぎ変数には、ALLOCATABLE、CODIMENSION、INTENT(OUT)、POINTER、VALUE 属性は含まれません。

ランク引き継ぎ変数は新しい構文 DIMENSION(..) で宣言されます。DIMENSION(*) にはすでに意味があるため、追加の意味と見なすことになります。繰り返しますが、仮変数にのみランク引き継ぎを含めることができます。スカラーを含む任意のランクの項目をランク引き継ぎ仮引数に渡すことができます (項目が型引き継ぎの場合、ランク引き継ぎ仮引数に渡す形状引き継ぎまたはランク引き継ぎになります)。ランク引き継ぎ仮引数のランクは実引数から引き継がれます (実引数がスカラーの場合は 0)。これらは記述子により渡されます (プロシージャーが BIND(C) の場合は C 記述子 - 下記参照)。ランク引き継ぎ仮引数には CODIMENSION 属性や VALUE 属性は含まれません。新しい RANK 組込み関数は引数のランクを返します。

2015 年 2 月の標準規格会議で、我々は、SELECT TYPE と似た概念の、新しい SELECT RANK 構造の提案を受理しました。この構造はランク引き継ぎ変数でのみ使用でき、各選択ブロック内では「結合変数」でランクを指定します。この件はまだ ISO 委員会では受理されていませんが、2015 年 8 月の会議で受理された話を聞けることを期待しています。SELECT RANK 構造なしでは、Fortran コードでランク引き継ぎ変数を使用したさまざまな処理を行うことは困難です。

C 記述子は Fortran 標準規格から派生したもので、形状、型、ALLOCATABLE/POINTER 属性などのプロシージャー引数に関する拡張情報を通信するための標準に準拠した方法を定義します。標準規格では、C 記述子のアクセスおよび操作用に、C 記述子の typedef、さまざまなメンバーの値のマクロ、関数の定義を宣言する、ISO_Fortran_binding.h という名前の C ヘッダーファイルを Fortran 実装で提供することを明示しています。記述子の変更は、これらの関数のいずれかを使用して行います。すべての typedef、マクロ、関数は「CFI_」で始まります。重要なポイントは、標準規格では C 記述子のレイアウトについていくつかの制限を定義していますが、実装依存のメンバーおよびメンバーの順序に余地が残されていることです。このため、C 記述子が異なる Fortran 実装間で必ず交換できるとは限りません。この状況でも、ベンダー拡張に依存しない、Fortran と C が混在するアプリケーションを作成することは可能です。

C 記述子を使用した Fortran-C プログラムの例を次に示します。

```
use, intrinsic :: iso_c_binding
interface
  function c_alloc (array) bind(C)
  import
  integer(C_INT) :: c_alloc
  real(C_FLOAT), intent(out), allocatable, dimension(:) :: array
  end function c_alloc
end interface
real(C_FLOAT), allocatable, dimension(:) :: my_array
if (c_alloc(my_array) == 0) then
  print *, lbound(my_array), ubound(my_array); print *, my_array
  end if
end
```

```
#include "ISO_Fortran_binding.h "
extern int c_alloc (CFI_cdesc_t * descr) {
    int ret, i; float * array;
    CFI_index_t lower = 0, upper = 10;
    ret = CFI_allocate (descr, &lower, &upper, 0); // No elem_len
    if (ret == CFI_SUCCESS) {
        array = descr->base_addr;
        for (i=lower;i<=upper;i++) {array[i] = (float) i;}
    }
    return ret;
}
```

「Coarray TS」

TS18508 に含まれている 2 つめの大きな機能セットは、一般に「Coarray TS」と呼ばれる、「Fortran での追加の並列化機能」です。「Interop TS」とは異なり、Coarray TS はまだ流動的です。昨年 11 月の時点では、私はすでに遠大になっていたスケジュールがその通りになるとは楽観視していませんでした。しかし、2015 年 2 月の会議で、最も時間がかかると思われていた部分に大きな変更が加えられたことで、スケジュール通りになる可能性が高くなったと考えています。

Coarray TS には 4 つの項目 (チーム、イベント、アトミック、コレクティブ) があります。「チーム」は、特定のタスクでともに動作する Co-Array アプリケーションのイメージのコレクションです。アプリケーションには、独立して動作し、親に結果を伝えるいくつかの (または多くの) チームが含まれます。チームの優れている点は、イメージの番号付けがチームに関連しているため、アプリケーション全体の「共通形状」について心配することなく、Co-Array を使用するライブラリーを簡単に構築できることです。追加された構文で、共通インデックス参照の一部としてチーム変数を指定できます。

チームは自身のサブチームを形成でき、チームは動的に消滅および再生成することができます。その理由の 1 つは、イメージの 1 つが失敗した場合です。イメージの数が多くなるとともに、この可能性もより多くなります。これは議論が白熱した部分でした。イメージが失敗したのか、応答に時間がかかっているだけなのかをどのように区別すれば良いのでしょうか?

当初、TS には、イメージの処理に失敗したがイメージが残されている「ストール」という概念が含まれていました。我々は、このトピックについて多くの会議で討論しましたが、ストールしたイメージを検出して回復する方法を言語で表現する解決方法を見つけ出すことはできませんでした。2 月に、我々はストールのアイデアをすべて捨て去ることにしました。代わりに、イメージが失敗したことを検出できるように、ステータス引数を同期構造に追加しました (もちろん、検出の実際のメカニズムは実装依存です)。ただし、2 月の会議では、通常発言を行う一部のメンバーが討論に参加していなかったため、改訂版の TS が 8 月の会議で受理されるかどうかはいまのところ不明です。

TS のもう 1 つの大きな変更点は、「イベント」(タスクが完了し続行可能になったことをほかのイメージに通知する方法) です。この部分は、しばらくの間変更されていません。私はこれ以上変更されるとは思っていません。

Fortran 2008 では操作を個別に行う ATOMIC オブジェクトの概念が追加されましたが、制限付きのサポートでした。Coarray TS では、ADD、AND、CAS (コンペア・アンド・スワップ)、OR および XOR のようなアトミック操作用のプロシージャーが追加されます。

コレクティブは、現在のチームのすべてのイメージで操作を行うための組込みプロシージャーです。新しく定義されたサブルーチンは、CO_MAX、CO_MIN、CO_SUM、CO_BROADCAST、CO_REDUCE です。

その他

上記の 2 つの技術仕様の追加に加えて、Fortran 2015 では、言語の一貫性を向上するためいくつかの小さな変更が行われました。その一部を次に示します。

- 配列コンストラクターおよび DATA で暗黙的な DO 変数の型および種別を指定することができます (この変更はユーザーフォーラムで提案されたものです)。
- SIZE= はアドバンシング入力 (Q 形式拡張子など) とともに使用できます。
- G0.d は整数型、論理型および文字型リスト項目とともに使用できます。
- 新しい組込み関数 RANDOM_INIT、COSHAPE、REDUCE、OUT_OF_RANGE が追加されました。
- IMPORT はプロシージャーおよび BLOCK で利用できるようになりました。
- IMPLICIT NONE(EXTERNAL) はすべてのプロシージャーで明示的なインターフェイスが必要です。
- すべてのプロシージャーはデフォルトで再帰的です。必要に応じて NON_RECURSIVE を指定します。

標準委員会は機能を追加することもあれば、機能を削除することもあります。ラベル付き DO ループ (DO 10 I…)、EQUIVALENCE、COMMON および BLOCK DATA は (非推奨) になりました。算術 IF および (DO の範囲が CONTINUE や END DO で終了しない) ノンブロック DO 構造は、(非推奨) になり、最終的に言語から削除される予定です。皆さんが使用するコンパイラーからこれらの機能がただちに削除されることはありませんが、これらの機能を使用した場合、標準規格に準拠しているかどうか確認したときに非準拠のフラグが付けられるでしょう。

調査の結果

最初に述べたように、私は SC14 セッションの出席者の皆さんに、Fortran の使用法に関する調査への参加を依頼しました。回答者はそれほど多くありませんでしたが (全部で 11 名)、興味深い結果が得られました。

- 45% は現在 Co-Array を使用中で、64% は 3 年以内に Co-Array を使用する予定でした。
- 73% は現在 C との相互運用機能を使用中で、36% は 3 年以内に DO CONCURRENT を使用する予定でした。
- 言語混在アプリケーションで使用されているほかの言語では C (90%) が圧倒的で、次点は Python (27%) でした。
- 回答者の 90% は仕事で 3 つ以上の異なる Fortran コンパイラーを使用していました。
- 回答者の Fortran プログラミングの年数は平均 14 年でした (最短 4 年、最長 28 年)。
- 回答者は全員 (100%)、5 年後も Fortran を使用していると考えていました。

この調査は、現在も受け付けています。フォームの URL は、http://goo.gl/forms/j55wX7EyPs (英語) です。

Fortran 2015 についてのコメントは、Blog または調査フォームにお寄せください。コンパイラーまたは Fortran 言語についてのヘルプが必要な場合は、ユーザーフォーラム (英語) で質問してください。Windows* ユーザー向けと Linux*/Mac* ユーザー向けのフォーラムがあります。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください