インテル® oneAPI ベース & HPC ツールキット

新しい Fortran コンパイラーへの移行に向けて

エクセルソフト株式会社

はじめに

- 本セミナーは下記の移行ガイドをもとにしています
  - Porting Guide for ifort Users to ifx (intel.com)
- コンパイラー・オプションは Linux、Windows 向けで異なる場合があります
  - オプション名の先頭は Linux だと - 、Windows は / です
    - Linux 例: -help
    - Windows 例: /help
  - Windows 環境において先頭の - は / と認識されます

※資料中のコンパイラー・オプションの多くは Linux 向けで表記しています

バージョン 2024 がリリースされました
2023年11月リリース

- 2024 のリリースにともないバージョン 2023.2.x は旧バージョンとなりました
  - Intel Registration Center や各種パッケージマネージャーからインストーラーをダウンロードできます
- 一部コンポーネントは個別にパッチがリリースされています
  - コンパイラーバージョン: 2024.0.2 など
- メジャーリリースとなるため API やランタイムに旧バージョンとの互換性がなくなる場合があります

インテル® oneAPI ベース & HPC ツールキット

コンパイラーや実行環境

- インテル® Fortran コンパイラー・クラシック
- インテル® Fortran コンパイラー
- インテル® oneAPI DPC++/C++ コンパイラー
- インテル® ディストリビューションの Python*

ツール/ライブラリー

- インテル® MPI ライブラリー
- インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)
- インテル® oneAPI DPC++ ライブラリー (インテル® oneDPL)
- インテル® oneAPI コレクティブ・コミュニケーション・ライブラリー (インテル® oneCCL)
- インテル® oneAPI マス・カーネル・ライブラリー (インテル® oneMKL)
- インテル® oneAPI ディープ・ニューラル・ネットワーク・ライブラリー (インテル® oneDNN)
- インテル® oneAPI データ・アナリティクス・ライブラリー (インテル® oneDAL)
- インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP)
- インテル® DPC++ 互換性ツール
- oneAPI ベース・ツールキット用 インテル® FPGA アドオン

解析/デバッグツール

- インテル® Inspector
- インテル® Trace Analyzer & Collector
- インテル® VTune™ プロファイラー
- インテル® Advisor
- インテル® ディストリビューションの GDB

個別ダウンロードが必要なコンポーネントを含みます

インテル® **HPC** ツールキット

インテル® oneAPI **ベース**・ツールキット

インテル® ソフトウェア開発ツールの Fortran コンパイラー

- oneAPI ツールキットのリリース以降、2種類のコンパイラーが提供されています
  - インテル® Fortran コンパイラー・クラシック
    - ifort コマンド
    - 2024/01/25 時点の最新バージョン: 2021.11.1
  - インテル® Fortran コンパイラー
    - ifx コマンド
    - 2024/01/25 時点の最新バージョン: 2024.0.2

インテル® Fortran コンパイラー・クラシックの非推奨

- バージョン 2024 のリリースに伴い
  インテル® Fortran コンパイラー・クラシックは非推奨になりました
- 2024年後半の製品パッケージから除外され
  サポートおよびダウンロード提供の終了が予定されています
  - 満了していない有償サポートをお持ちの方は
    最大 2025 年までダウンロードできます
- パッチバージョン 2024.0.1 において ifort でコンパイル時に
  非推奨を知らせるメッセージを出力するようになりました
- インテルは ifx への移行を推奨しています
- macOS 向け ifort は 2024.0 より削除されています

ご紹介内容

- インテル® Fortran コンパイラーの概要
- ifx 機能紹介
- ifx への移行に向けて
  - 補足: Visual Studio* 統合環境 (Windows ユーザー様向け)

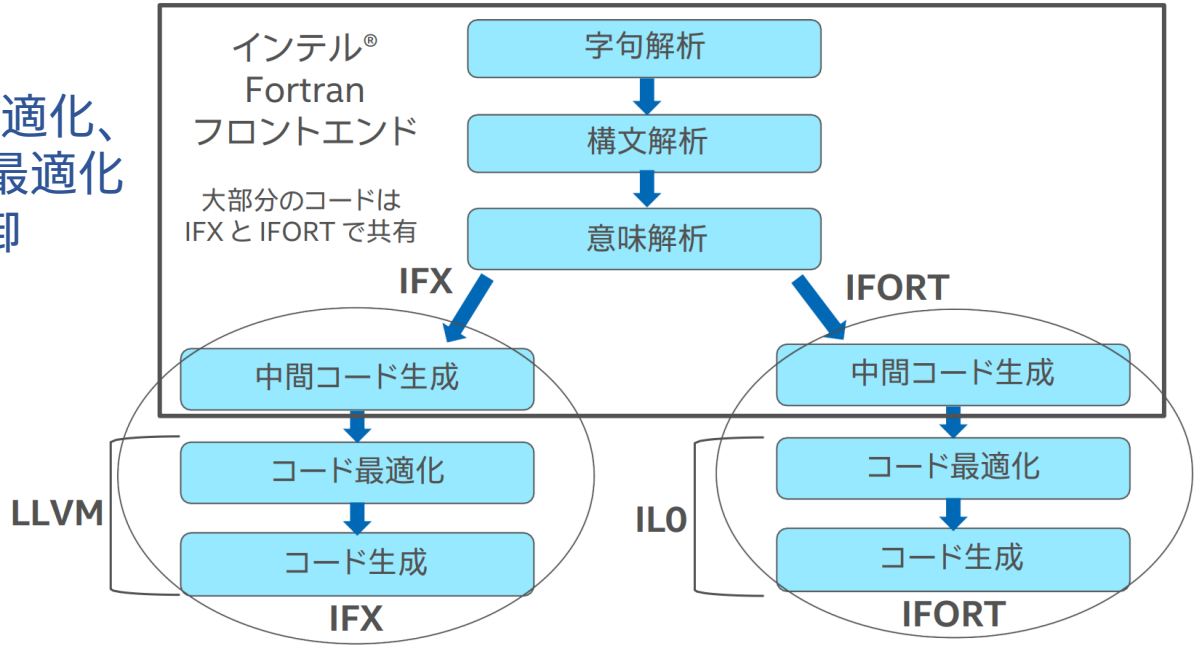
インテル® Fortran コンパイラー (ifx)

- 新規開発された新しい Fortran コンパイラー
- **ifort フロントエンド** と **LLVM バックエンド** を採用
- インテル® oneAPI ツールキット 2021 よりベータ版が提供開始されバージョン 2023 に正式リリースされました

ifort と ifx の比較

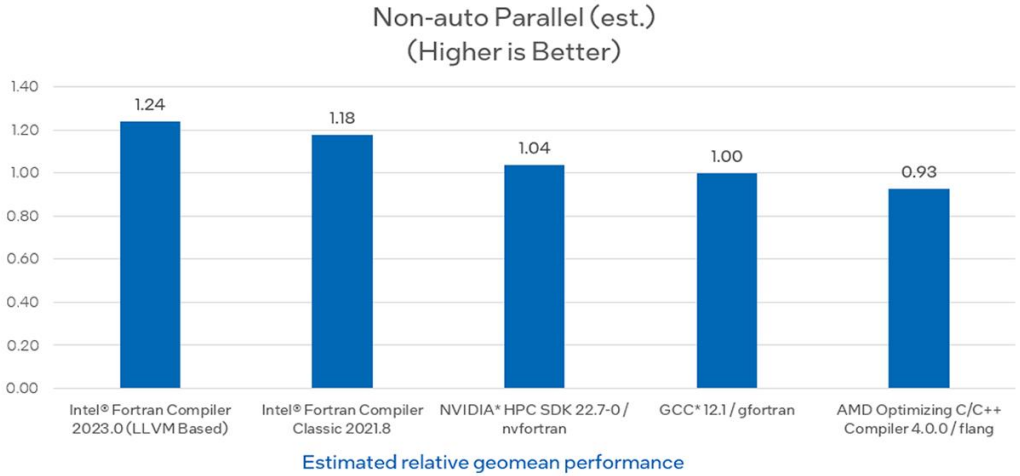
- 最適化やコード生成における基盤が異なります

自動ベクトル化
プロシージャー間の最適化、
プロファイルに基づく最適化
浮動小数点演算の制御
最適化レポート
インライン展開
アンロール
etc...



Intel® Fortran Compiler Boosts Application Performance on Linux*

Performance Advantage Measured by Polyhedron* Fortran Benchmark on Intel® Core™ i9-12900K Processor



Testing Date: Performance results are **based on testing by Intel as of December 2, 2022** and may not reflect all publicly available security updates.

Configuration Details and Workload Setup: Intel® Core™ i9-12900K CPU @ 5.2GHz, i9-12900K, 16G x2 DDR5 4800. Software: Intel® Fortran Compiler for applications running on Intel® 64, Version 2023.0.0 Build 20221201, Intel® Fortran Compiler Classic for applications running on Intel 64, Version 2021.8.0 Build 20221119_000000, GCC 12.1.0 / gfortran, AMD* Optimizing C/C++ Compiler 4.0.0 / flang - AMD clang version 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28) (based on LLVM Mirror.Version.14.0.6), NVIDIA* HPC SDK 22.7-0 / nvfortran. Red Hat Enterprise* Linux release 8.4 (Ootpa), 4.18.0-372.9.1.el8.x86_64. Non-auto Parallel compiler switches: Intel® Fortran Compiler: -Ofast -xalderlake -flto -nostandard-realloc-lhs. Intel® Fortran Compiler Classic: -fast -xCORE-AVX2 -nostandard-realloc-lhs. GCC / gfortran: -Ofast -mfpmath=sse -flto -march=alderlake -funroll-loops. AMD* Optimizing C/C++ Compiler / flang: compile: -O3 -ffast-math -march=znver3 -fveclib=AMDLIBM -flto -mllvm -unroll-aggressive -mllvm -unroll-threshold=500; link: -Wl,-mllvm -Wl,-inline-recursion=4 -Wl,-mllvm -Wl,-lsr-in-nested-loop -Wl,-mllvm -Wl,-enable-iv-split -flto -Wl,-mllvm -Wl,-region-vectorize -Wl,-mllvm -Wl,-function-specialize -Wl,-mllvm -Wl,-align-all-nofallthru-blocks=6 -Wl,-mllvm -Wl,-reduce-array-computations=3 -O3 -ffast-math -march=znver3 -fveclib=AMDLIBM -flto -lamdlibm -lflang -lamdlibm -lm. NVIDIA HPC SDK / nvfortran: -fast -Mipa=fast,inline -Mallocatable=03 -Mfprelaxed -Mstack_arrays.

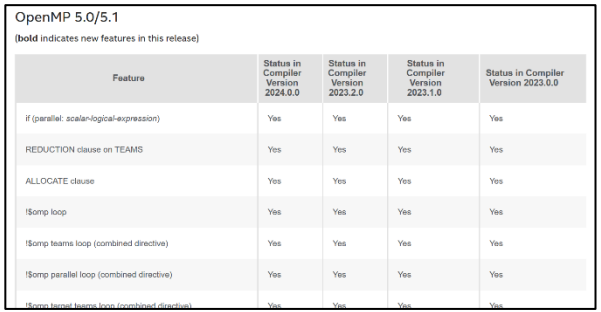
Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

[ベンチマーク：出典]

ifx の言語規格、OpenMP* サポート

- Fortran 言語規格
  - 77, 90, 95, 2003, 2008, 2018 サポート
    および 2023 の一部を実装　(ifort は 2018 の一部サポートまで)
    - 2023 のサポート例:
      - DO CONCURRENT REDUCE
      - BOZ 定数の機能拡張
- OpenMP* 規格
  - OpenMP* 4.5 以前、5.0、5.1、5.2、6.0
    - Intel® Fortran Compiler Fortran Language and OpenMP* Features
  - インテル® GPU 向け TARGET 構文のサポート



OpenMP 5.0/5.1

(bold indicates new features in this release)

Feature	Status in Compiler Version 2024.0.0	Status in Compiler Version 2023.2.0	Status in Compiler Version 2023.1.0	Status in Compiler Version 2023.0.0
if (parallel: *scalar-logical-expression*)	Yes	Yes	Yes	Yes
REDUCTION clause on TEAMS	Yes	Yes	Yes	Yes
ALLOCATE clause	Yes	Yes	Yes	Yes
!$omp loop	Yes	Yes	Yes	Yes
!$omp teams loop (combined directive)	Yes	Yes	Yes	Yes
!$omp parallel loop (combined directive)	Yes	Yes	Yes	Yes
!$omp target teams loop (combined directive)	Yes	Yes	Yes	Yes

ifx の利用

- ifx はインテル® HPC ツールキットに同梱されています
- ifx の呼び出し
  - Visual Studio* 統合環境
  - コマンドライン

```
> ifx source.f90
```

    - ifort 同様に環境設定が必要です
      Linux* : setvars.sh もしくは oneapi-vars.sh
      Windows* : Intel oneAPI command prompt からコマンドプロンプトを起動

ifx コンパイラー・オプション

- インテル固有の最適化
- LTO (IPO)、PGO
- 最適化レポート
- サニタイザー
- OpenMP*

インテル固有のプロセッサー向け最適化

- x,-ax、/Qx, /Qax オプションの指定がない場合、
  LLVM 標準による最適化とベクトル化を適用します
  - インテル固有の最適化を指示

```
> ifx -O2 -xcore-avx2 source.f90
```

  - AVX512 における 512bit レジスタの利用には
    –mprefer-vector-width=512 を使用します

x,-ax、/Qx, /Qaxオプションは
コンパイラーのヒューリスティックにより
指示した命令セットを使用しない場合があります

- 最新のプロセッサー世代のサポートが提供されています
  - 例:AVX512_FP16 命令などのサポートを含む
    sapphirerapids キーワードを引数に指定できます

```
> ifx -O2 -xsapphirerapids source.f90
```

x86 アーキテクチャー向け最適化

- -mauto-arch=<value>、/Qauto-arch=<value>
  - パフォーマンスの利点がある場合にx86 アーキテクチャーをベースとしたプロセッサー向けに複数の機能固有の自動ディスパッチ・コードを生成するように指示します
  - value は x,-ax、/Qx, /Qax オプションの指定と同じ

```
> ifx /Qauto-arch=CORE-AVX2 source.f90
```

  - x,-ax、/Qx, /Qax と一緒に使用できません
    - 最後に記述した方が優先されます

IPO、PGO

- IPO、PGO はそれぞれ LLVM が提供する LTO、PGO 機能に置き換えられました
  - LTO - Link Time Optimization
  - PGO - Profile-Guided Optimizations

LTO

- -flto、/flto または -flto=[arg]、/flto=[arg]
  - arg = thin、full (デフォルト)

```
> ifx /O2 /Qxcore-avx2 /flto /fuse-ld=lld source.f90
> ifx -O2 -xcore-avx2 -flto source.f90
```

  - ThinLTO は full LTOと比較してメモリ消費量を抑えつつビルド時間の短縮を目指した異なる LTO の手続きを提供します
    - 詳細は ThinLTO — Clang 18.0.0git documentation (llvm.org)
- -ipo および /Qipo は LTO を呼び出します
- Windows では /flto とあわせて /fuse-ld=lld を追加で指定します
  - MSVC の link リンカーの代わりに lld リンカーを使用する必要があります

PGO

- LLVM の PGO 実装
  - インストルメント方式もしくは外部プロファイラーによるサンプリング
- インストルメント方式
  - -fprofile-instr-generate、-fprofile-instr-use の組み合わせ
- llvm-profdata ツールの別途ダウンロードが必要です
  - llvm パッケージに付属

-fprofile-ml-use, /fprofile-ml-use

- コンパイラーの静的ヒューリスティックに機械学習による訓練済みモデルを使用します

```
> ifx -fprofile-ml-use source.f90
```

  - インストルメンテーション方式の PGO で最適化される分岐予測の改善が期待できます

最適化レポート

- LLVM ベースの YAML ファイルへの出力 (コミュニティベースの機能)
  +
  従来のテキストファイルのレポート出力 (インテルの機能)
- -qopt-report=<n>, /Qopt-report:<n>
  - <n> = 1~3, 指定ない場合は 2

```
> ifx -O2 -xcore-avx2 -qopt-report=2 source.f90
```

- *.yaml もしくは *.optrpt ファイルから最適化レポートを確認できます
  - YAML ファイルを確認する場合は別途 llvm-opt-report ツールを利用
    llvm-opt-report - generate optimization report from YAML — LLVM 19.0.0git documentation

*.optrpt 最適化レポート

```
LOOP BEGIN at ./matmul.f90 (26, 13)
            remark #25566: blocked by 64
            remark #25563: Load hoisted out of the loop
            remark #15300: LOOP WAS VECTORIZED
            remark #15305: vectorization support: vector length 8
            remark #15475: --- begin vector loop cost summary ---
            remark #15476: scalar cost: 83.000000
            remark #15477: vector cost: 14.281250
            remark #15478: estimated potential speedup: 5.796875
            remark #15309: vectorization support: normalized vectorization overhead 0.125000
            remark #15488: --- end vector loop cost summary ---
            remark #15447: --- begin vector loop memory reference summary ---
            remark #15450: unmasked unaligned unit stride loads: 32
            remark #15451: unmasked unaligned unit stride stores: 16
            remark #15474: --- end vector loop memory reference summary ---
            remark #25583: Number of Array Refs Scalar Replaced In Loop: 12
LOOP END
```

-fsanitize=<sanitizer>
/fsanitize=<sanitizer>

- LLVM のサニタイザー機能を有効にして、
  実行時に特定の問題を検出できます
- <sanitizer> キーワード
  - address
    - バッファー・オーバーフロー/アンダーフロー
    - メモリーリーク(Linuxのみ)
  - memory (Linux のみ)
    - 初期化されていない変数。(-check uninit の指定と同じ動作)
  - thread (Linuxのみ)
    - スレッド間のデータ競合
    - OpenMPプログラムを実行する場合は環境変数
      「OMP_TOOL_LIBRARIES='libarcher.so'」の設定を推奨

OpenMP* スレッドのデータ競合

```
> ifx -fiopenmp -fsanitize=thread -g -O2 *.f90
> ./a.out
```

```
26  !$OMP PARALLEL DO
27  do i=1, n
28    x = h * (DBLE(i)-0.5d0)
29    sum = sum + f(x)
30  end do
31  !$OMP END PARALLEL DO
```

```
WARNING: ThreadSanitizer: data race (pid=1185457)
  Write of size 8 at 0x7fffa30eabe8 by thread T15:
    #0 pi_calc_.DIR.OMP.PARALLEL.LOOP.232.split36 /home/xlsoftkk/work/pi_calc/pi-calc.f90:28:5 (a.out+0x4f039e) (BuildId:
a25fdbb986fd385726123d7a165267d46a0c4e57)
    #1 __kmp_invoke_microtask <null> (libiomp5.so+0x167d92) (BuildId: 0b92a2ad5b16d95f07791fdede39499bf1d23cea)

  Previous write of size 8 at 0x7fffa30eabe8 by main thread:
    #0 pi_calc_.DIR.OMP.PARALLEL.LOOP.232.split36 /home/xlsoftkk/work/pi_calc/pi-calc.f90:28:5 (a.out+0x4f039e) (BuildId:
a25fdbb986fd385726123d7a165267d46a0c4e57)
    #1 __kmp_invoke_microtask <null> (libiomp5.so+0x167d92) (BuildId: 0b92a2ad5b16d95f07791fdede39499bf1d23cea)
    #2 pi_calc_.void /home/xlsoftkk/work/pi_calc/main.f90 (a.out+0x4eee0c) (BuildId: a25fdbb986fd385726123d7a165267d46a0c4e57)
    #3 MAIN__ /home/xlsoftkk/work/pi_calc/main.f90:16:18 (a.out+0x4eee0c)
    #4 main <null> (a.out+0x4346ac) (BuildId: a25fdbb986fd385726123d7a165267d46a0c4e57)

  Location is stack of main thread.

  Location is global '??' at 0x7fffa30cb000 ([stack]+0x1fbe8)

  Thread T15 (tid=1185473, running) created by main thread at:
    #0 pthread_create <null> (a.out+0x43c51f) (BuildId: a25fdbb986fd385726123d7a165267d46a0c4e57)
    #1 __kmp_create_worker <null> (libiomp5.so+0x169587) (BuildId: 0b92a2ad5b16d95f07791fdede39499bf1d23cea)
    #2 pi_calc_.void /home/xlsoftkk/work/pi_calc/main.f90 (a.out+0x4eee0c) (BuildId: a25fdbb986fd385726123d7a165267d46a0c4e57)
    #3 MAIN__ /home/xlsoftkk/work/pi_calc/main.f90:16:18 (a.out+0x4eee0c)
    #4 main <null> (a.out+0x4346ac) (BuildId: a25fdbb986fd385726123d7a165267d46a0c4e57)

SUMMARY: ThreadSanitizer: data race /home/xlsoftkk/work/pi_calc/pi-calc.f90:28:5 in pi_calc_.DIR.OMP.PARALLEL.LOOP.232.split36
```

OpenMP* オプション

- OpenMP* によるスレッド並列を有効にするために
  -fiopenmp もしくは /Qiopenmp オプションを指定します
  - -qopenmp、/Qopenmp と同等です

```
> ifx -fiopenmp source.f90
```

- target ディレクティブによる GPU へのオフロード機能が利用できます
  - -fopenmp-targets、/Qopenmp-targets オプション
    - fopenmp-targets = spir64

```
> ifx -fiopenmp -fopenmp-targets=spir64 source.f90
```

- -qopenmp-simd は O1 以上の最適化レベルで有効化されます

spir64: 64-bit Standard, Portable Intermediate Representation

インテル製の GPU へオフロード

- target 領域の処理を GPU へオフロードするように指示します

```
> ifx -fiopenmp -fopenmp-targets=spir64 source.f90
```

  - ホストから GPUデバイスへデータをマップ
  - target 領域の実行中ホスト側は待機します

```
!$omp target map(to: a, b) map(tofrom: c )
!$omp parallel do
do j=1,N
    do i=1,N
        do k=1,N
            c(i,j) = c(i,j) + a(i,k) * b(k,j)
        enddo
    enddo
enddo
!$omp end parallel do
!$omp end target
```

DO CONCURRENT の自動オフロード

- -fopenmp-target-do-concurrent
  /Qopenmp-target-do-concurrent
- 他のオフロード向けオプションと組み合わせます

```
> ifx -fopenmp-target-do-concurrent -fiopenmp -fopenmp-targets=spir64 source.f90
```

```
INTEGER,DIMENSION(N) :: J, K
INTEGER              :: I, M
M = 10
I = 15
DO CONCURRENT (I = 1:N, J(I)> 0)LOCAL (M) SHARED (J, K)
   M =  MOD (K(I), J(I))
   K(I) = K(I) - M
END DO
PRINT *, I, M
```

DO CONCURRENT を
OpenMP* の target 領域に変換します

ifx への移行に向けて

ifx コンパイラー・オプション

- ifort がサポートしているコンパイラー・オプションの多くは
  そのまま ifx でも受け付けるように設計されています
- 一方でバックエンドが異なることから
  同じオプションでも動作や実装の違いがあります
  - 同じオプションを指定していても、実行時間や計算結果の違いが
    発生する可能性があります

ifx からサポートを終了したオプション

- ifx にてサポートされなくなった ifort オプションを
  記述すると警告を出力します
  - 例：自動並列化 (-parallel、/Qparallel)、32bit プログラムの生成 (/Qm32、-m32)

```
ifx: command line warning #10148: option '/Qparallel' not supported
```

- サポートされているオプション、および
  サポートを終了したオプションは、
  -qnextgen-diag
  /Qnextgen-diag
  を指定すると一覧を表示できます

```
options being removed:
    /G3
    /G4
    /G5
    /G6
    /G7
    /GB
    /MP-force
    /Qauto-ilp32
    /Qcilk-serialize
    /Qcomp-obj
    /Qconditional-branch
    /Qcov-gen
    /Qemit-build-options
    ...
```

移行に向けて

- ifort コマンドを ifx へ変更
  - 新しい定義済みマクロ __INTEL_LLVM_COMPILER により ifx のみバージョンを取得できます (Cmake でも利用可能)
    - ifort は __INTEL_COMPILER を利用
- -O2 –xCORE-AVX2 といったシンプルな組み合わせから試してみる
  - 実行時やコンパイル時にエラーが表示される場合は いったん ifx の最適化機能を無効にしてみる
- コンパイラー・オプションの対応
- 浮動小数点数演算における結果の再現性

最適化の無効とデバック向け情報出力

```
> ifx -g -O0 -warn all -traceback -qno-openmp-simd -standard-semantics source.f90
```

- 最適化の無効化
  -O0、/Od
- コンパイル時の警告を確認する
  -warn all　　/warn:all
- ランタイムチェック
  -check all、/check:all
- 実行時エラーのトレースバックを出力
  -traceback、/traceback
- デバッグ用のシンボルを作成する
  -g、/debug:full
- Fortran 言語仕様に準拠した動作
  -standard-semantics、/standard-semantics
- OpenMP* SIMD ディレクティブの無効化
  -qno-openmp-simd、/Qopenmp-simd-

主なコンパイラー・オプションの対応

- 自動ベクトル化機能を活用する
  - -x,-ax、/Qx, /Qax オプションを指定する (-Ofast も可)
    - AVX512 命令セットにおける 512bit 幅のレジスタを利用するためには -qopt-zmm-usage=high の代わりに –mprefer-vector-width=512 オプションを使用します
- PGO
  - -fprofile、-Qfprofile オプションの手続きを利用ください
  - もしくは分岐予測の最適化を目的に -fprofile-ml-use オプションで代替します
- IPO
  - -flto、/flto オプションへ変更 (任意)
  - インテルが提供していたリンカーツール xilink、xild、および xiar は削除されています
- OpenMP*
  - OpenMP* による SIMD サポートを無効化するには -qno-openmp-simd、/Qopenmp-simd- を指定ください

主なコンパイラー・オプションの対応

- 自動並列化機能 (-parallel、/Qparallel)
  - 並列領域を OpenMP* を使用して
    明示的にコンパイラーへ指示する必要があります
    - !DIR$ omp parallel do
  - ifort の最適化レポートオプションを使用することで
    コンパイラーによって並列化されていた領域の
    OpenMP* 指示句を確認できます
    - -qopt-report-phase=par、/Qopt-report-phase:par

```
> ifort /Qparallel /Qopt-report-phase:par /Qopt-report:5 source.f90
```

```
remark #17109: LOOP WAS AUTO-PARALLELIZED
   remark #17101: parallel loop shared={ } private={ } firstprivate={ K J ? A ? B ? C 15 I } lastprivate={ }
firstlastprivate={ } reduction={ }
```

主なコンパイラー・オプションの対応

- 最適化レポート
  - -qopt-report=<n>、/Qopt-report=<n> のみ使用できます
  - 現時点では *.optrpt レポートの利用が推奨されています
    - *.yaml ファイルのレポートも生成しますが、インテルが提供するベクトライザーによる最適化情報が反映されません

浮動小数点演算の再現性

- -fp-model <type>, /fp:<type>
  - ifx は <type> に precise、fast (デフォルト)、strict のいずれかを指定できます
- ifort で機能していた一部の<Type>は利用できません
  - consistent と同等の動作を ifx に指示する場合は -fp-model precise -fimf-arch-consistency=true -no-fma を指定します

```
> ifx -fp-model precise -fimf-arch-consistency=true -no-fma *.f90
```

- fast キーワードは ifort と ifx で動作が異なることに注意ください
  - ifx は浮動小数点比較に NaN オペランドをチェックしません。ifort と同等の動作を指示する場合は、-assume nan_compares を指定します

算術ライブラリーにおける関数の精度

- -fimf-precision=<value>、/Qimf-precision:<value>
  - ライブラリー関数における精度のレベルを変更します
  - <value> = high、medium (デフォルト)、low
- -fimf-arch-consistency=<value>、/Qimf-arch-consistency:<value>
  - 異なるマイクロ・アーキテクチャー間で一貫した結果を生成します
- ifort ではそれぞれ関数名による指示ができますが、ifx ではできません
- その他、浮動小数点数演算に関わるオプションは下記をご参考ください
  - Floating-Point Options (intel.com)

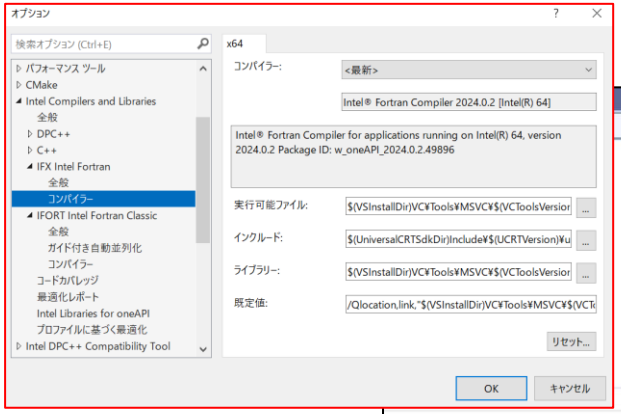
互換性

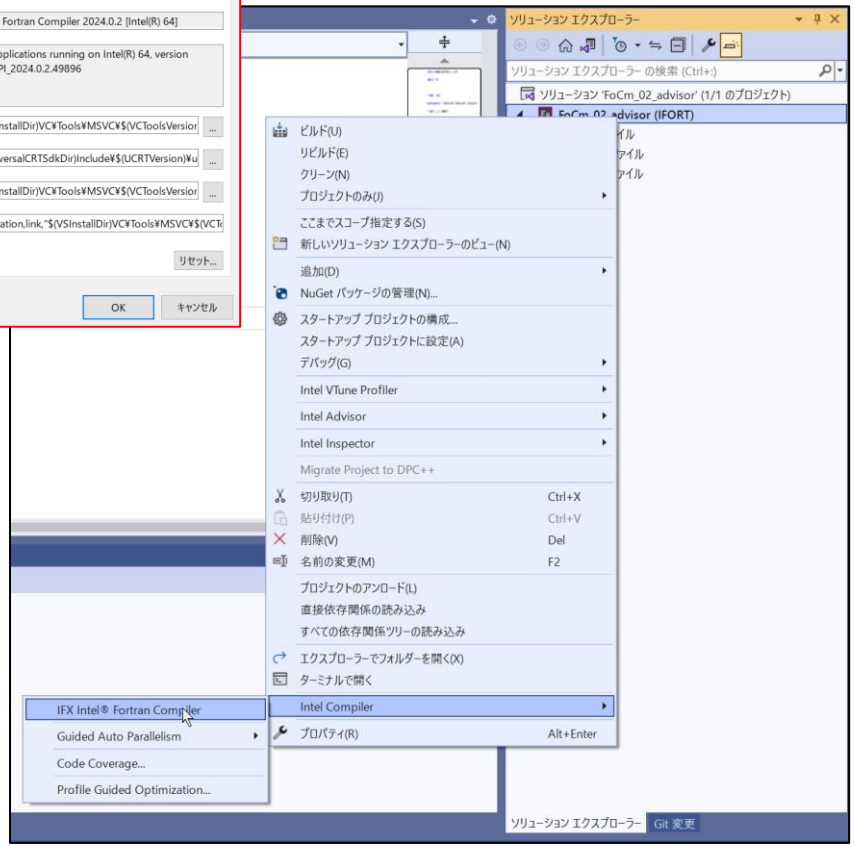
- ifx で生成されたオブジェクトファイル、モジュールファイルは ifort と互換性があります
  - バイナリーとライブラリーは、ifx、ifort でそれぞれリンクが可能であり、ほかのコンパイラーで生成された .mod ファイルも使用できます
- icc、icl、icpc、icx、icpx、dpcpp で作成されたオブジェクトファイルもバイナリ互換のためリンクできます
- IPO や LTO によって作成されるオブジェクト・ファイルは互換性がありません
- いずれも 64bit プログラムに制限されます

Visual Studio* 統合

- ifx においても ifort 同様に統合環境上で利用できます
- インテル® ソフトウェア開発ツールのインストール前に
  サポートされる Visual Studio* 環境を事前にインストールください
  - Microsoft* Visual Studio* 2019
  - Microsoft* Visual Studio* 2022
- Community、Enterprise、Professional Edition への
  統合がサポートされます コンパイラーと IDE の互換性 (intel.com)
- コンパイラーバージョン 2024 から Visual Studio* の
  64bit プログラム構成の Fortran プロジェクトに設定される
  デフォルトのコンパイラーは ifort から ifx に変更されました

ifort ⇔ ifx 切り替え





- x86 構成のプロジェクトに
  対して ifx へ切り替えると
  自動的に x64 構成に設定されます
- コンパイラーバージョンの選択は

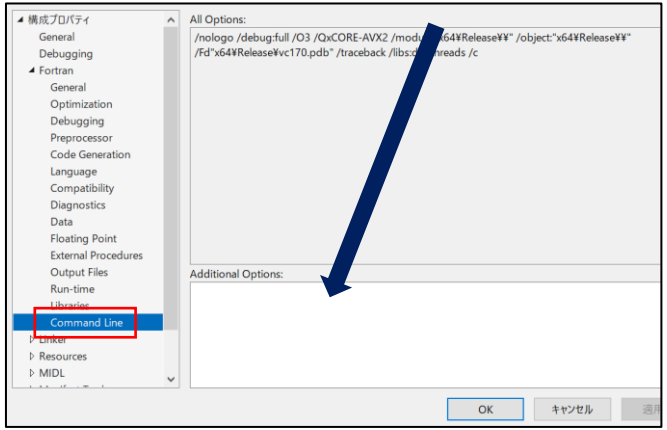
  [ツール]>[オプション]>[Intel Compiler and Libraries]

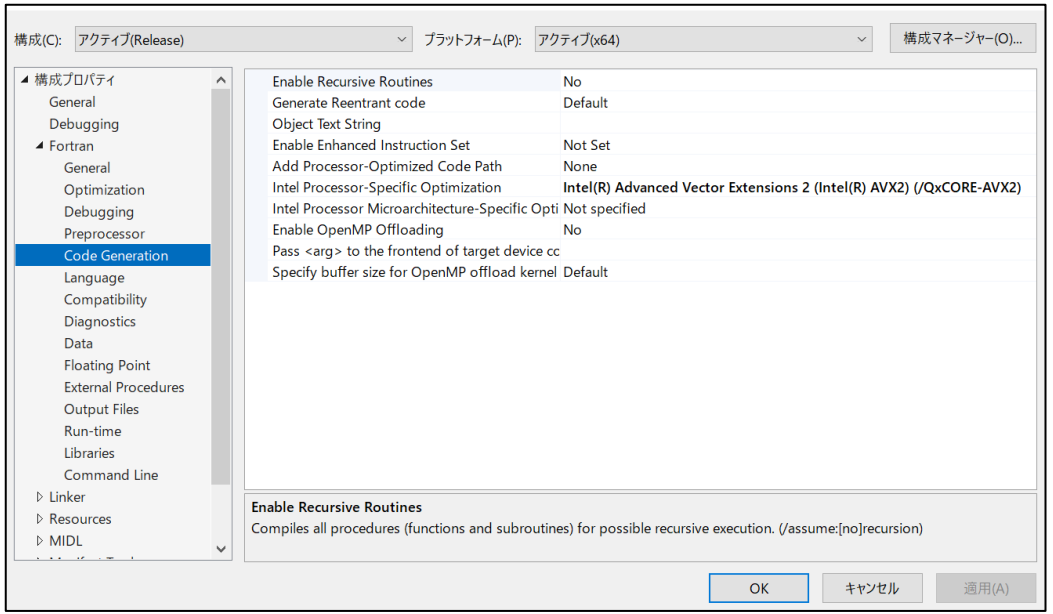
  から ifx および ifort それぞれ設定できます

プロジェクト・プロパティ

- 各コンパイラー・オプションはプロジェクト・プロパティから設定ください

項目が存在しないオプションは
Command Line 項目から直接入力します





補足情報

- 日本語情報
  - インテル® oneAPI ポーティング・ガイド (ifx) (isus.jp)
  - LLVM リンク時の最適化設計と実装 | iSUS
- 英語情報
  - New Features for ifx Only (intel.com)
- llvm-profdata や llvm-opt-report の入手先
  - Releases · llvm/llvm-project (github.com)

ifort の継続利用について

- ブログ記事
  - Deprecation of The Intel® Fortran Compiler Classic (ifort) - Intel Community
- メンテナンスおよびアップデートの終了により
  セキュリティアップデートを含む更新がなくなります
- コンパイル時に表示される非推奨を示すメッセージを消すには
  -diag-disable=10448、/Qdiag-disable:10448 を指示します