フローチャート自動生成ツール「yFlowGen」の使い方

目次

1　はじめに

本文書はフローチャート自動生成ツール「yFlowGen」に関する使用マニュアルです。

2　本ツールの機能

yFlowGen.exe に C 言語(C,C++)のソースファイルを入力することでフローチャートを自動生成します。
ファイル形式は、「GML ファイル」、「DOT ファイル+SVG ファイル」を選択可能です。

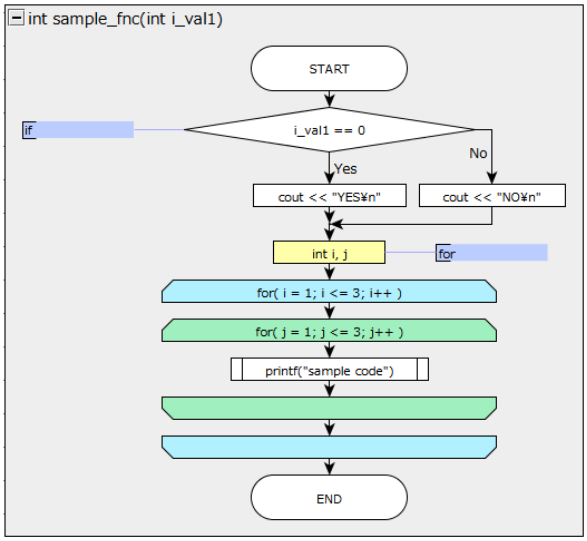
- GML ファイルの場合、表示・整列するために yEd Graph Editor というソフトのインストールが必要です。
- DOT+SVG ファイルの場合、DOT を SVG ファイルに変換するために Graphviz というソフトのインストールが必要です。DOT+SVG ファイルの場合は Graphviz の機能により自動整列も行われます。

[実行例] GML ファイル生成の場合

下記コマンドを windows のコマンドプロンプトで実行。例ではソースファイルとして sample.c を指定。

```
yFlowGen.exe -f sample.c
```

```
int sample_fnc(int i_val1) {
    /* if */
    if (i_val1 == 0) cout << "YES¥n";
    else
        cout << "NO¥n";
    // for
    int i, j;
    for (i = 1; i <= 3; i++){
        for (j = 1; j <= 3; j++){
            printf("sample code");
        }
    }
}
```

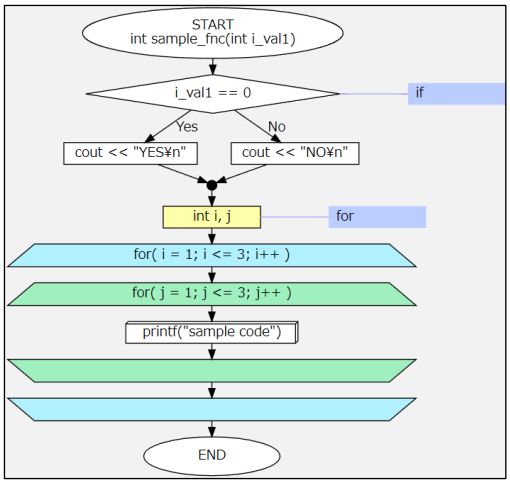


[実行例] DOT+ SVG ファイル生成の場合

オプションに「-format dot」を追加することで DOT ファイル生成可能となります。

```
yFlowGen.exe -f sample.c -format dot
```

```
int sample_fnc(int i_val1) {
    /* if */
    if (i_val1 == 0) cout << "YES¥n";
    else
        cout << "NO¥n";
    // for
    int i, j;
    for (i = 1; i <= 3; i++){
        for (j = 1; j <= 3; j++){
            printf("sample code");
        }
    }
}
```



3　yFlowGen.exe の使い方

3.1　yFlowGen.exe の実行方法

Windows のコマンドプロンプトにて、yFlowGen.exe を実行することで、result_yFlowGen フォルダ以下にフローチャートのファイルを出力し、log_yFlowGen.txt に実行ログを出力します。実行の際は、下記の引数を指定してください。

```
yFlowGen.exe -f <filePath> -no_compact -no_comment -out1file -out_group_comment
もしくは
yFlowGen.exe -d <dirPath> -no_compact -no_comment -out1file -out_group_comment

-f <filePath> : C 言語で書かれたソースファイルのパス （-d を記載の場合は省略可能）
-d <dirPath>  : C 言語で書かれたソースファイルを含んだフォルダのパス （-f を記載の場合は省略可能）
-format <format> : 「-format dot」と入力すると DOT と SVG ファイルを生成します。（省略時は GML を生成）
-out1file     : 1 ソースファイルにつき 1GML ファイルを出力する （省略可能）
-no_compact   : 処理ブロックのサイズをできるだけ小さくする設定を OFF （省略可能）
-no_comment   : フローチャートにコメントを表示しない （省略可能）
-no_color     : ブロックに色を付けない （省略可能）
-no_dec       : 宣言のみのブロックは表示しない(省略可能)
-out_group_comment : コメントをグループの外側に配置する(省略可能)
-no_disp_struct : struct, union のフローは出力しない(省略可能)
-no_reset     : スクリプト実行時に結果フォルダを削除しない(省略可能)
-true_false   : if/else if 文の真偽を True,False で記載する(未設定時は Yes,No で記載)
-no_connection_point : 接続点を追加しない(未設定時は接続点を追加)
-define <defA,defB,defC=1,etc>: 有効な#define 名 （省略可能. 複数指定の場合はカンマ区切りで指定）
-disp_invalid_def : 無効な#define 記述をコメントとして表示(省略可能)
-left_flow_is_no : IF 分岐の左を NO、右を YES に設定 （省略可能）
-add_extension <cxx,cs,etc> : 追加の拡張子 （省略可能）
-pj_name <your_project_name> : プロジェクト名 （省略可能）
-copy_org_code : 入力ファイルのソースコードを result フォルダにコピーする （省略可能）

実行例:
 yFlowGen.exe -f sample.c -out1file
```

- -f の後にファイル名を指定して実行すると、yFlowGen.exe を実行した場所に result_yFlowGen¥ソースファイル名"のフォルダを作成し、関数ごとにフローチャートのファイルを出力します。
- -d の後にフォルダ名を指定して実行すると、指定したフォルダ以下にある C 言語で書かれたソースファイル(.c, .cpp)を検索し、result_yFlowGen フォルダ以下に、検索したフォルダの階層構造と同じ階層構造のフォルダを作成し、そこへフローチャートのファイルを出力します。
- C コード内で#define を使用する場合は、-define の後ろにスペースを入れて、有効な#define 名をカンマ(,)区切りで記載してください。（例：-define AAA,BBB,CCC=1,DDD=-1）
- yFlowGen は結果の一覧を「result_yFlowGen.html」「result_yFlowGen¥index.html」に出力します。
  また、-pj_name <プロジェクト名>を用いてプロジェクト名を指定すると、result_yFlowGne(フォルダ、html)を<プロジェクト名>_result_yFLowGen として生成します。-pj_name オプションを使用しない場合は、result_yFlowGen を生成します。

なお、コマンドプロンプトでの実行方法だけでなく、GUI で実行する方法も別途用意しています。
(→yFlowGenGUI について)

3.2 制限事項

- 「-f」、「-d」で指定するパスに日本語を含めると動作しない場合があります。このため、パスに日本語は入れないでください。
- 「printf や cout などの”で囲まれた箇所、およびコメント」以外に日本語を含めると文字コードの問題から正しく動作しない場合があります。もし変数等に日本語を使用したい場合は日本語を「”」で囲んでください。
  (例：for(”カウント“=0;”カウント”<100;”カウント”++) など)
- try～catch 例外処理において、図を見やすくするために catch の最終ブロックと次のブロックとを接続するようにしています。
- enum のフローチャートは出力しません。
- class でのメンバ関数の表現はループブロックを用いて表現しています。
- スコープに関しての表現はループブロックに「SCOPE」と記載して表現しています。
-

3.3 GML ファイルを生成する場合の設定

3.3.1 GML ファイルは yEd Graph Editor を使って開きます

生成したファイルは gml ファイル形式になっており、yEd Graph Editor を使用して開いてください。yEd Graph Editor は下記から無料でダウンロードできます。yEd Graph Editor は素晴らしいグラフエディタですので、ぜひ手でフローチャートを書く際もお使いください。

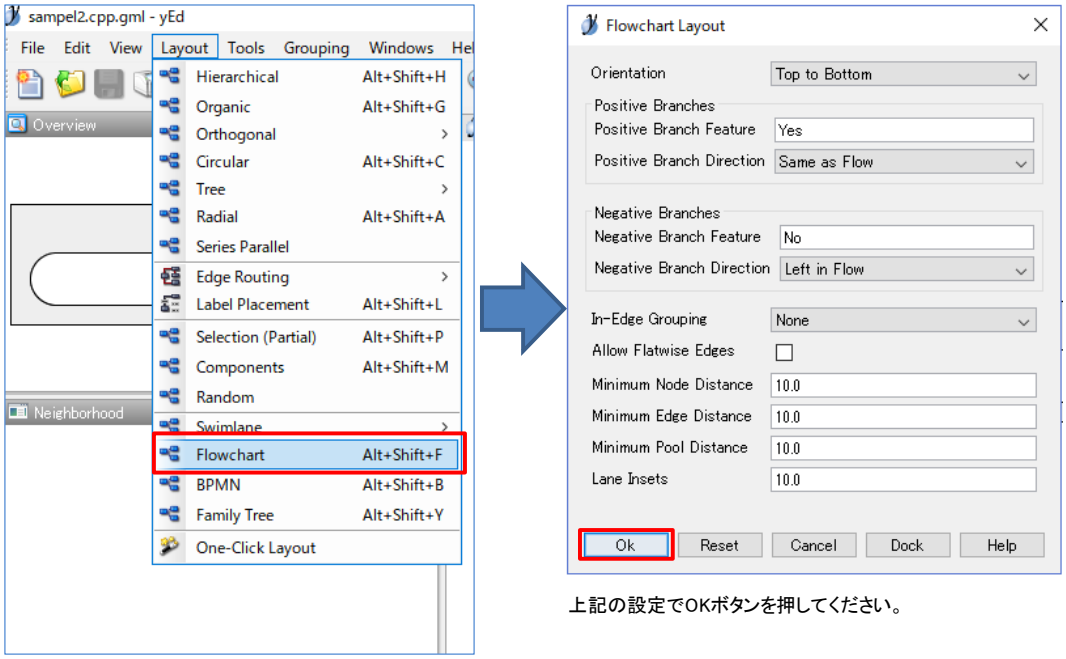
【[yWorks] yEd Graph Editor - Downloads】

https://www.yworks.com/downloads#yEd

3.3.2 yEd Graph Editor でのフローチャートの自動整列

gml ファイルを開いたら、yEd Graph Editor の自動整列機能を使用し図を自動整列します。

ツールバーの Layout→Flowchart にて設定・実行ウィンドウを開いて下記設定で実行してください。



上記の設定でOKボタンを押してください。

3.4　DOT+SVG ファイルを生成する場合の設定

3.4.1　DOT ファイルを使用する場合 Graphviz が必要です

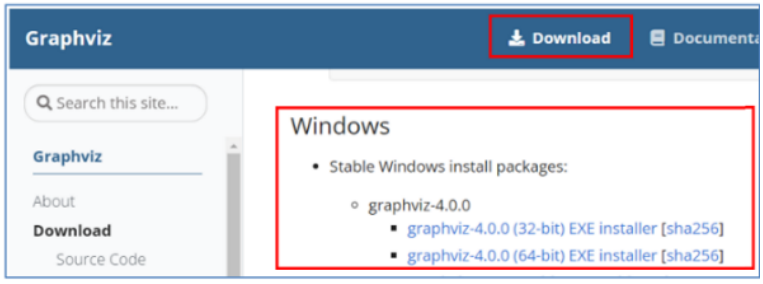
DOT ファイル生成する場合予め Graphviz をインストールしておく必要があります。

yFlowGen では、生成した DOT ファイルを Graphviz を使用して画像ファイルである SVG ファイルに変換します。

Graphviz は下記からダウンロード可能です。

Windows10 をお使いの場合は、Stable package の 64bit 版をご使用ください。

【Download | Graphviz】https://graphviz.org/download/



Graphviz のインストーラは PC によってはセキュリティによりブロックされることがあります。

その場合は、インストーラを右クリックしてプロパティを開き、下記赤枠の「許可する」に✓をいれてください。

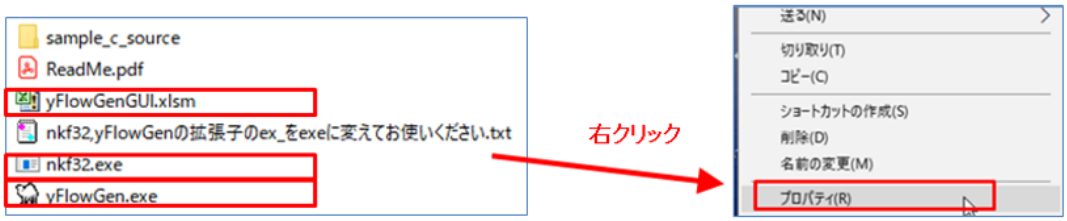
4　yFlowGenGUI について

yFlowGenGUI は Excel マクロシート上で yFlowGen.exe を実行することのできる補助ツールです。
詳しくは、本ファイルと同階層に置かれている「yFlowGenGUI.xlsm」の「使い方」シートをご覧ください。

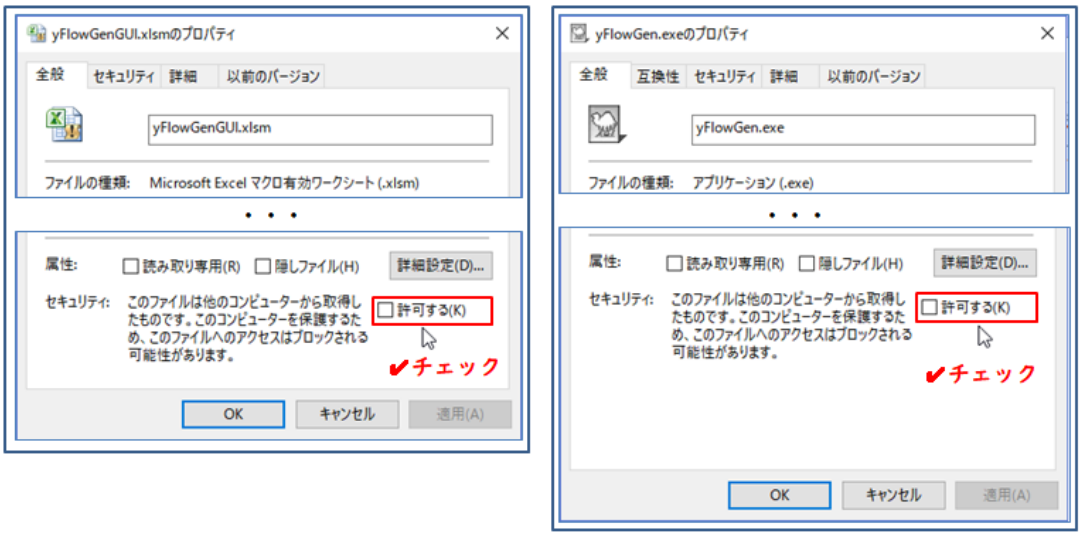
4.1　ブロック解除

Windows によりインターネットから取得したファイルはブロックを解除しないと実行できないようになっています。このため、下記の方法でブロック解除してください。（コマンドプロンプトからだとブロック解除しなくても実行できるようです）

【手順 1 】ファイルを右クリックしてプロパティを押してください



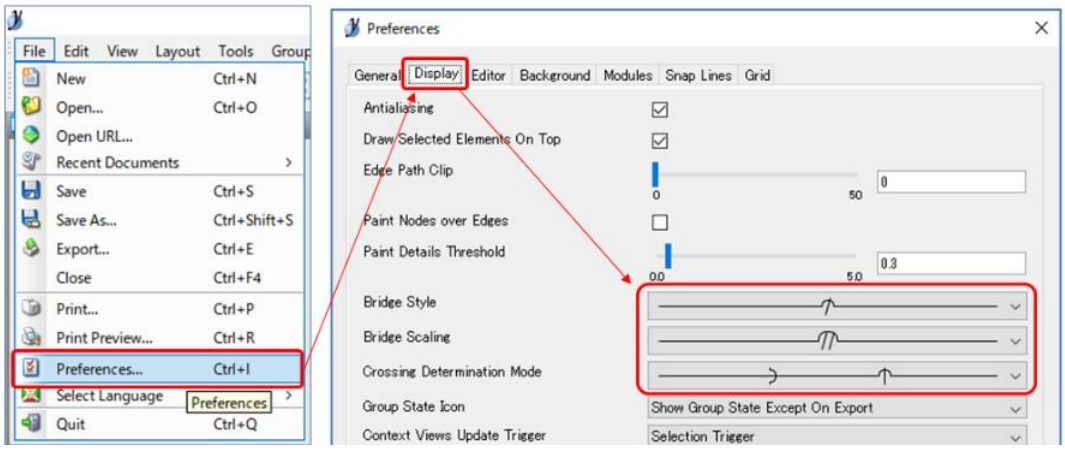
【手順 2】セキュリティの項目の「許可する」のチェックボックスにチェックを入れてください。
(yFlowGen.exe, yflowGenGUI.xlsm, nkf32.exe に対して必要です)



5 [yEd Graph Editor] より見やすい図にするための設定

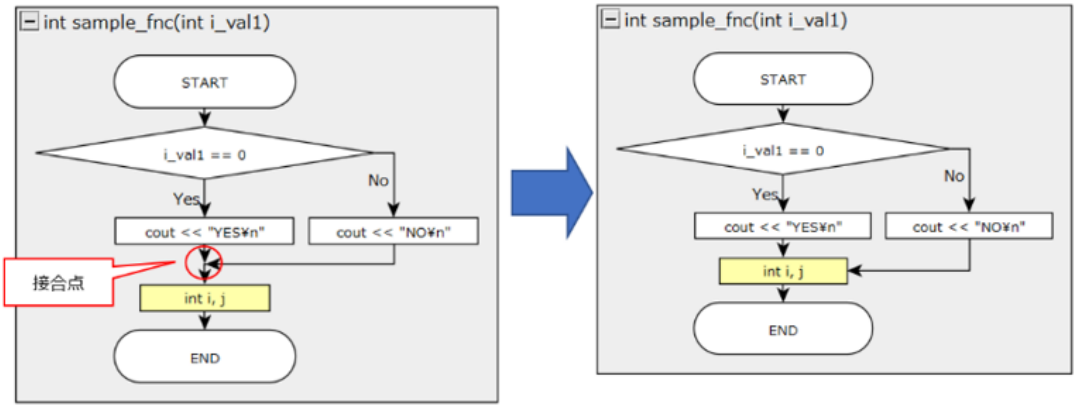
5.1 Bridge Style 設定

yEd Graph Editorの「Preferences」→「Display」タブのBridge Styleにて線が交差した時の表現を選べます。
最初の設定はブリッジ表現しない設定になっているためこちらを設定すると見やすくなると思います。



5.2 接合点を追加しない場合の設定

接合点を追加せず表示したい場合、「接合点を追加しない」設定(-no_connection_point)を ON してください。



「-no_connection_point」をOFFした場合(通常設定)　　「-no_connection_point」をONした場合

6 nkf32.exe について

nkf32.exe は下記 URL から取得し再配布しています。 nkf32.exe を用いることで入力ファイルの多様な文字コードに対応しています。yFlowGen.exe と同じ階層に置いて使用してください。

【nkf.exe : Vector 】https://www.vector.co.jp/soft/dl/win95/util/se295331.html

(作成者 URL : http://hp.vector.co.jp/authors/VA007219/)

7 動作環境

・OS: windows7(32bit 版/64bit 版)以上を推奨 (windos10 64bit 版での動作確認済)

8 使用条件

このプログラムはフリーウェアです。著作権は、toowaki が持ちます。

転載は自由に行ってください。また、当ソフトおよび、添付の Excel シートの改ざん・変更等を行わないようお願いします。

9 免責

このプログラムを使用して生じた損害等につきましては、作者はいっさい関与しません。

使用者の責任で、本プログラムを使用してください。

10 連絡先

何かご要望等ございましたら、toowaki.fc2@gmail.com までメールお願いします。

11 履歴

日付	内容	Ver.	編集者
2018/01/05	新規作成	1.0	toowaki
2018/05/02	「printf(“text”);}」という行(printf 限定)で正しく処理されていなかったため修正しました。	1.1	
2018/11/08	下記 3 点において正しく動作しておらず修正しました。 ・インデントのための“}”が同じ行で連続している場合 ・同じ行で”;”の直後に”}”がある場合 ・else 文を 1 行で記載し、”{“を使用しない記述を行った場合	1.2	
2018/11/23	・printf, cout 内に”(”や”)”がある場合に正しく動作しておらず修正しました。 ・同じ行で”}“の直後に制御文(if,else など)がある場合に正しく動作しておらず修正しました。 ・制御文でない 3 行以上の 1 文で正しく動作しない場合があったため修正しました。 ・「スコープの最後かつ、関数の最後でかつ、インデントが 2 つ以上下がる場合」に対応できていなかったため修正しました。	1.3	
2018/12/02	・「既知の問題(Known Issue)」を記載した章を ReadMe 文書に追加しました。 ・中カッコを用いた配列代入に対応できておらず修正しました。	1.4	
2018/12/08	・for 文を一行で記載し、かつ処理を中カッコで囲っている場合に対応しておらず修正しました ・for(;;)の記載が 2 行以上で記載されている場合には対応できておらず修正しました。	1.5	
2018/12/29	・中カッコのみのスコープに対応しました。 ・class, struct, namespace 等に対応しました。	2.0	
2019/01/02	・nkf32.exe を用いて入力ファイルの多様な文字コードに対応しました。 ・if/else if 文の真偽を True, False で記載する設定を追加しました。	2.1	
2020/06/14	・break 文との接続に誤りがあり修正しました。	2.2	
2020/08/23	・switch 文の「default:」と同じ行に処理記載がある場合にその記載がフローチャートに反映されていなかったため、修正しました。	2.3	
2020/08/30	・for, while, do-while のループ内で break を使用し、そのループ内にさらにループがある場合にそのループを出たところと先ほどの break を接続していたため、修正しました。	2.4	
2021/06/26	・switch 文の各 case での処理後、break を改行せずに同じ行に記述すると break を通常処理として扱っていたため、修正しました。	2.5	

日付	内容	Ver.	編集者
2021/07/07	・「else;」という記載に対して正しく処理していなかったため修正しました。	2.6	
2021/07/10	・else 文の処理が空となる場合に対応しました。	2.7	
2022/02/12	・If 文の特定条件(if 文内の最終行に else 文で終わらない if/else if 文を記載し、かつ、その後処理文なく終わる場合)において正しく接続されていなかったため、修正しました。	2.8	
2022/03/21	・オプション「接合点を追加しない」を追加しました。	2.9	
2022/04/10	・break 文との接続に誤りがあり修正しました。	2.10	
2022/07/03	・Graphviz を用いた DOT+SVG ファイル生成に対応しました。 ・生成結果の一覧を「result_yFlowGen.html」として生成するようにしました。	3.0	
2022/09/11	・オプション「-ignoredef」(無効な#define 名の指定)を追加しました。	3.1	
2022/09/18	・オプション「-disp_invalid_def」(無効な#define 記述をコメントとして表示)を追加しました。	3.2	
2023/02/12	・オプション「-left_flow_is_no」(IF 分岐の左を NO、右を YES に設定)を追加しました。	3.3	
2023/02/18	・拡張子が大文字の場合も読み込むよう対応 ・オプション「追加拡張子」を追加	3.4	
2023/07/02	・exit 関数, _exit 関数を return 関数と同じように扱い、end ブロックに接続されるようにしました。 ・#ifndef に対応できていなかったため、#ifndef に対応しました。 ・#define 名の指定方を「無効な define 名の指定」(-ignoredef)から「有効な define 名の指定」(-define)に変更しました。 ・#if 0 ～#endif, #if 1 ～#endif に対応しました。	4.0	
2023/07/18	・オプション「-pj_name」(プロジェクト名)を追加しました。 ・1 行の中で「;」を用いて複数の処理が記載される場合に対応しました。 (例えば、右記が 1 行で記載されるなど、「if (a) sub01(); else sub2();」)	4.1	
2023/07/22	・「if 条件の処理が空かつ、その if-else 条件が終わった直後にネストが 1 段以上浅くなる場合」に正しく接続できていなかったため、修正しました。 また、本動作を確認するために、sample_c_source¥sample_if.c を追加しました。	4.2	
2023/08/09	・#define-#else の表示方法を変えました。 ・#define-#else、#if-#elif-#else の接続について改善しました。 ※#if-#elif の条件判定にて演算子を用いる場合については対応しておらず、常に#else を選択するようにしてあります。今後のバージョンでの本機能の対応を検討しています。	4.3	
2023/08/15	・#if-#elif の条件判定にて演算子を用いる場合について対応しました。	4.4	
2023/11/04	・オプション「-copy_org_code」(入力ファイルのソースコードを result フォルダにコピー)を追加しました。これにより、別環境に result フォルダのみ移動し開いてもコードも参照できます。	5.0	