データとして何が必要かの洗い出しをする。

OPをやるにあたって、何のデータが必要となるかの種類を

区分分類分け

－通貨で分けて管理する

－

素材保管庫として、そのデータを置いてあるか。

何個あって、

基本的な

－何のデータを

－何個取るのか

－どういう形で

それがそろっていない場合にズレ

①何を使うのか

　－どういう形式の

　－何の素材を使うのか

　－どの情報をどういう形で使うのか決まっているか

・使用する値の形式

★何をどういう形でとってくるのかを決める

何を

どの形式で

何個

・何処からとってくるのか

　－その目的は、

　　　リアル、リアルのバックテスト、バックテスト

　－何個取らないといけないのがあって、

　　何個取れていて

　・合致したものが、合致した数で、ちゃんとデータとして取れているかをとったのか。

・どの形で保存する。

・どこから取りに行くのか。

・入る元データは一緒かどうか確認する

・計算した結果のデータが一致するか確認する

・元の条件（何が入ってくる必要があるのか）

　データのインポートまで来たら、

－素材がちゃんと取れているか

－ちゃんと処理ができているか

－ちゃんと出力できているか

　※引っ張ってくるものの条件はおのずともともとのインポート側のデータの制約によって、絞られる。

■確認の基準

・素材が正しい

・処理が正しい

・出力が正しい

　その時に持ってくる素材の条件が一致しているか

その状態の中でパッケージの作り方が決まっている

－設計の部分があるので、

　どういう風にくみたいか

　⇒なんのエンジンが必要

　　何のパッケージが必要

　　その条件はどうなっている

素材は処理する際の階層と一致している。

必要なパッケージ数は全て出る。

・おかしなデータが出ているかのチェック（足何本分のデータがあるのか）

・ルートで吐き出される数値は何と何と何と決まっているので、その数の部分で

・システム改修には、全一致でずれている箇所に対して、一括修正をシステム的に欠ける。

・システムの正常動作しているかを動かしている際に、どの階層まで正しく出ているかの

　チェックをする

・確認用のやつと、

　－素材の欠損

　－時間の欠損

　－入ってくるデータの

・もともとのシステムの担保が順番に取れているため、その手前で担保が取れているはず。

　・

・表現方法が決まっている

・置き換え対象がいくつあって、それに伴う置き換え数がいくつあるのか。

・データの管理方法として、

　置き換えが発生したなら、それだけ抜き出して置き換え処理ができるように

－元のやつと置き換えたやつ

　－AND条件が何個入っている、OR条件が何個入っているで変更前後の比率が

　　どう変わっているから、正しいのか、おかしいのかわかる。

・チェック項目は同じだから、

　変えたところでその数だけ変わる。

・こういうところをチェックしたいを確認項目を追加していく。

　⇒　リアルに関しては、それが走らないように作る。

・他通貨要素のリストが元データ部分にあれば、それを読みに行っているかだけ見れれば、OK

・置き換えする数の特定

・置き換え時の分類の変化の比率の確認

・

・素材から何が導き出される

・どこのエンジンをチェックします

・構成要素とパッケージをひっぱてこれる

■チェック用エンジン

・パッケージのエンジン

・素材のチェック

・片方はAで書かれている、片方はBで書かている　⇒　どこがずれているが出る

●入力時のチェック内容

－取得対象データリスト

－何のデータを

－何個取るのか

－どういう形で

－LogicSignal

　・数：条件名変換の行数分

　・形式：上下計算

　・分類：時間、通貨

　－Indicatorvalue

・数：nesl\_setting.csvに基づく

　・形式：数値（整数、小数）、日時（yyyy/mm/dd hh:mm）

　・分類：時間、通貨

　－Value（基本パッケージ）

　　・数：jikagaki.csvに基づく

　　・形式：数値（整数、小数）、日時（yyyy/mm/dd hh:mm）

　　・管理構成：ターゲットとルート番号と値

　・分類：時間、通貨、周期（0,1,5,6,9）

システムチェック内容

－値の有無（空データがないか）

－各フォーマットの枠に収まっているか

　・生価格（100.xxx円、1.xxxドル）

　・計算価格（100.xxxxxxxx円、1.xxxxxxxxドル）

　・日時（yyyy/mm/dd hh:dd:ss）

　・整数（n）

　・小数（n.nnnnnnnn）

　・文字列

　※何に分類されるのかがコード上でわかる形式の管理体制をとる（data\_format\_list.csv）

　　カラム構成：コード、データ形式番号、データ形式（メモ）

－参照先がない場合は、パッシブでエラー出力をする（何がない）

－とってくる必要データ数と、とってこれたデータ数の比較（rowとcolumnの数とconfigの内容の照合）

－とってくる場所の管理リスト

●処理のチェック内容

・元データ種類

・元データ数

・関数種類

・関数設定タイプ

－チェック結果を出力する（正常、異常いずれも）

　※異常の場合は、エラーコードを出力する

　ホワイト判断方式でパッケージタイプ、T/F、階層、データ形式の構成で判断ルール表をまとめる

●出力データ形式

出力値の比較

出力値のズレの親階層の抽出

●プログラムコメント

①判別処理

//引数：チェック対象の文字、チェック基準コード

//返値：true/false

//判定結果保持変数（デフォルト:true）：ans

//チェック対象文字のチェック位置管理用変数：count

//\*フラグ用変数：free\_flag

//{}フラグ用変数：count\_flag[]

//チェック用文字列1区画分：str\_check

//for\_チェック基準コードの文字列を1文字ずつ繰り返す

//if\_\*の場合

//flag\_\*フラグを立てる

//else if\_?の場合

//確認対象文字列の位置を1文字分ずらす（count++）

//str\_checkに?を格納

//flag\_\*フラグを外す

//else if\_#の場合

//str\_checkに#を格納

//if\_チェック対象文字列のcount位置の文字が数字であるかチェックする（数字である）

　//ans\_true

//else\_数字でない

//ans\_falseでブレークする

//flag\_\*フラグを外す

//------------------------

//else if\_{の場合

//現在位置からチェック基準コード内の}の文字を探索する

//{}内の文字列を抽出する

//抽出した文字列をstr\_checkに格納する

//“,”で文字列を分割する

//if\_分割した文字列の分割数が2区画の時

//else\_分割した文字列の分割数が

//------------------------

//else if\_[の場合

//現在位置からチェック基準コード内の]の文字を探索する

//[]内の文字列を抽出する

//ループ用変数を]の文字位置の数字に置き換える

//flag\_\*フラグを外す

//if\_!が文字列内に含まれる場合

//fort抽出文字列の分繰り返す

//if\_-の場合

//現在の文字の1文字前と1文字後の文字を確認する

//if\_1文字前の文字から1文字後の文字の中にチェック対象文字が含まれる場合

//文字の探索はあらかじめ数字、アルファベットの全文字列を別途作って起き、そこから開始文字と終了文字含めた文字列の抽出を行い、その文字列内にチェック対象文字が含まれるか確認する

//ans\_false ブレークする

//elseの場合

//現在の文字とチェック対象の文字が合致する場合

//ans false ブレークする

//else\_!が文字列内に含まれない場合

//fort抽出文字列の分繰り返す

//if\_-の場合

//現在の文字の1文字前と1文字後の文字を確認する

//if\_1文字前の文字から1文字後の文字の中にチェック対象文字が含まれない場合

//文字の探索はあらかじめ数字、アルファベットの全文字列を別途作って起き、そこから開始文字と終了文字含めた文字列の抽出を行い、その文字列内にチェック対象文字が含まれるか確認する

//ans\_false ブレークする

//elseの場合

//現在の文字とチェック対象の文字が合致しない場合

//ans false ブレークする

②処理プロセス

//引数：チェック対象の出力値、チェック基準設定ファイル

//返値：true/false

//---パッケージの出力値に対して評価する---

//---ホワイト判断方式でパッケージタイプ、T/F、階層、データ形式の構成で判断ルール表をまとめる---

//変数\_現在判定階層保持用：layer\_hold=0

//変数\_現在TF保持用：tf\_hold=false;

//変数\_パッケージタイプ合致保持用：package\_hold=false;

//対象ルートIDのパッケージタイプを取得する

//for\_process\_data\_type\_listの行を繰り返す

//if\_パッケージタイプが合致する

//package\_holdをtrueにする

//if\_layerがlayer\_holdよりも小さい場合ブレークする

//layer\_holdを現在のlayerの値で上書きする

//if\_出力結果がデータ形式と合致する場合

//tf\_holdに現在のtrue/flaseを上書きする

//else\_出力結果がデータ形式と合致しない場合

//なにもしない

//else\_パッケージタイプが合致しない

//if\_package\_holdがtrueの時ブレークする