レスバグ・プログラミング覚書

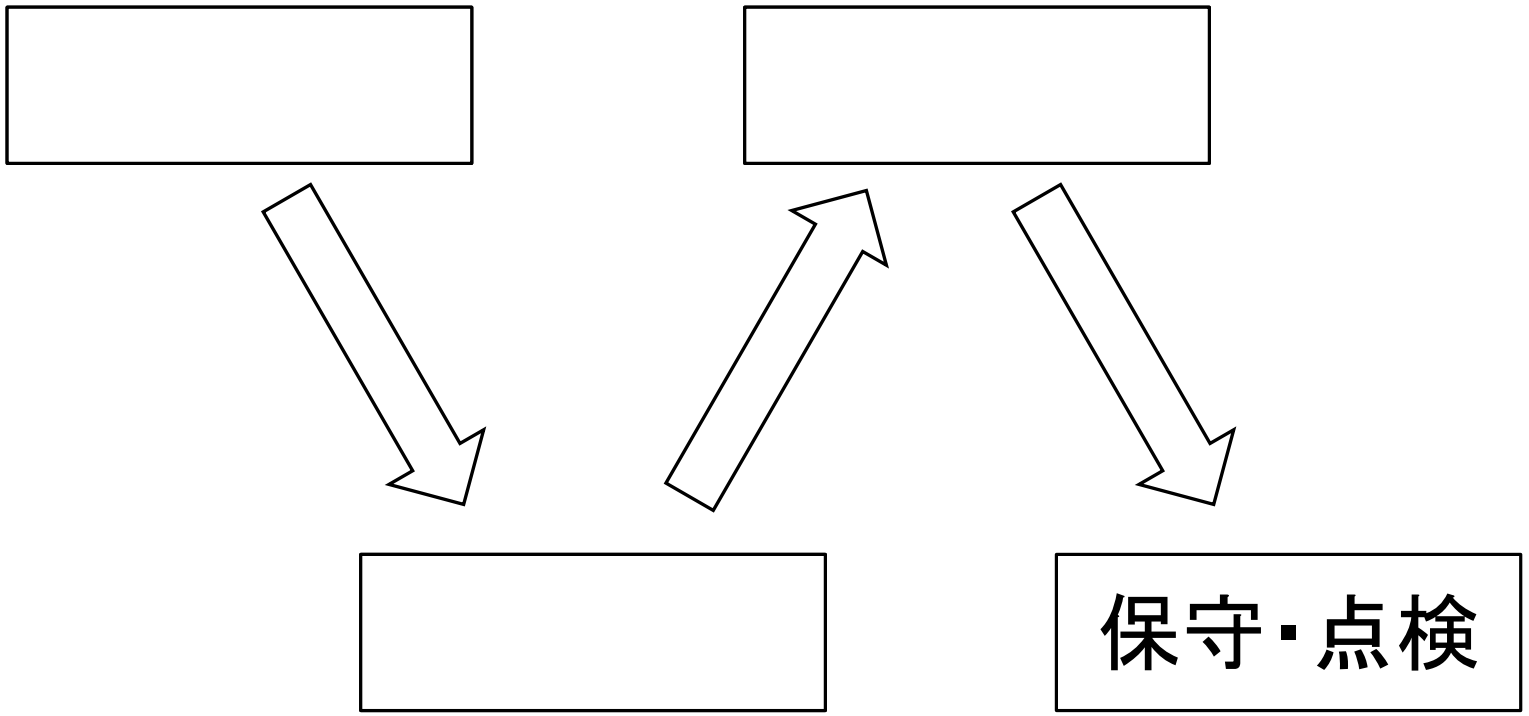
37-166278　村松克俊
（’16年度プログラミングTA）

皆さんは，

もの作りのルール

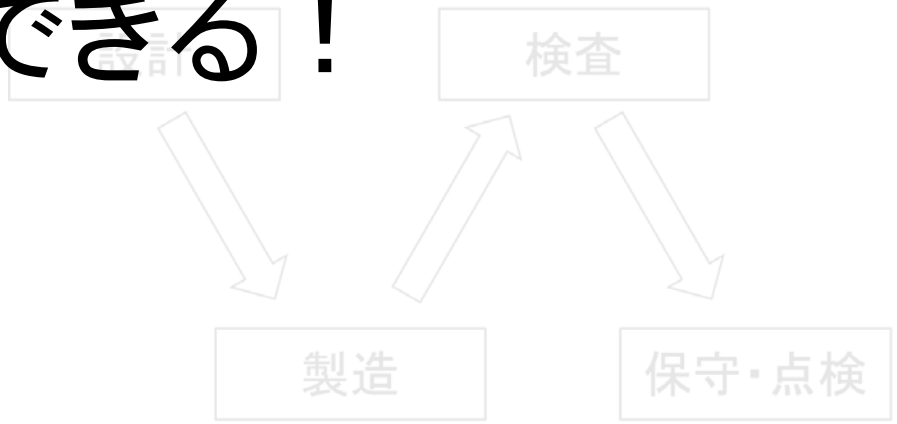
ってどのくらいご存知ですか？

一般的なもの作り



一般的なもの作り

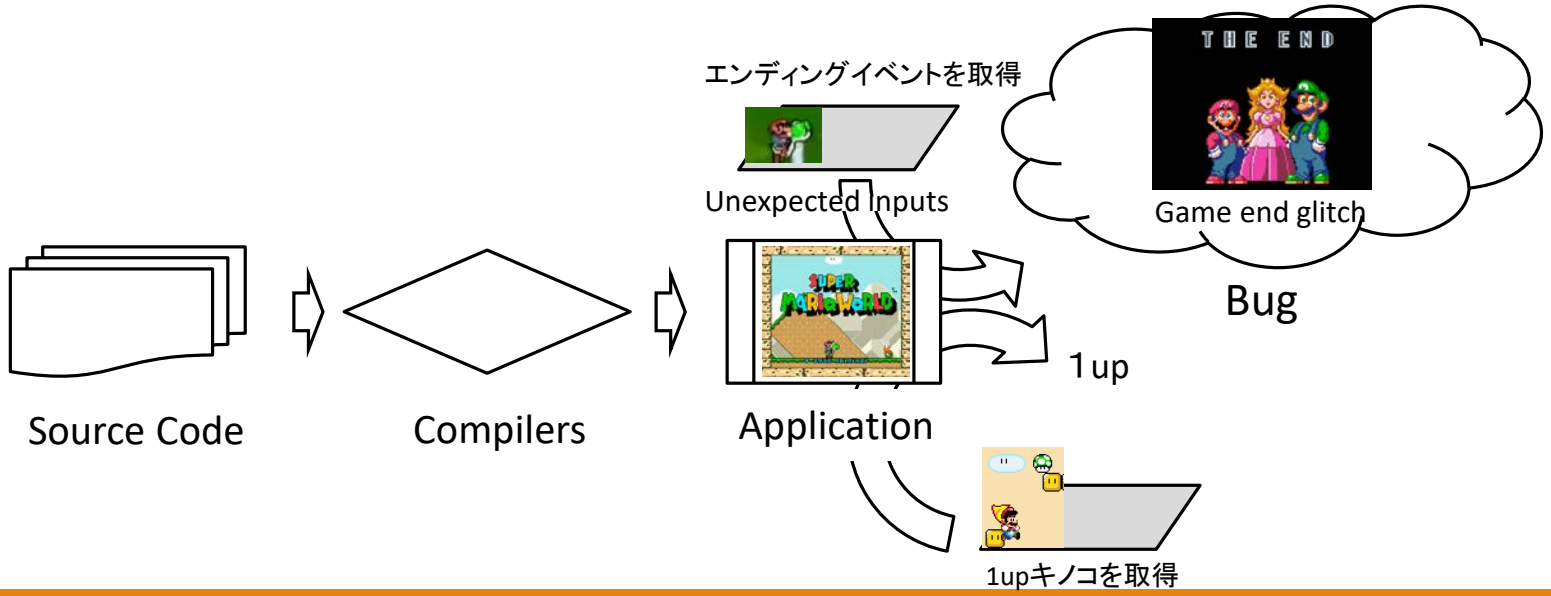
型に則れば,
バグの少ないプログラミングが
簡単にできる！



デバッグとは？

デバッグ(Debug)とは

- バグ（Bug）：不適切なソースやコンパイルの失敗などで発生した規定外の動作
- デバッグ（Debug）：Bugを取り除く作業全般



デバッグ三箇条

1. 予想を信じるな，結果のみを信じろ
2. デバッグは設計から始まる
3. 面倒な作業は決して厭わない

デバッグの手順

Step1
- バグの存在を認識する

Step2
- バグの発生条件を特定する

Step3
- バグの原因を特定する

Step4
- バグを駆逐する

Step5
- 修正を試験する

バグの存在を認識する

バグの発生要因

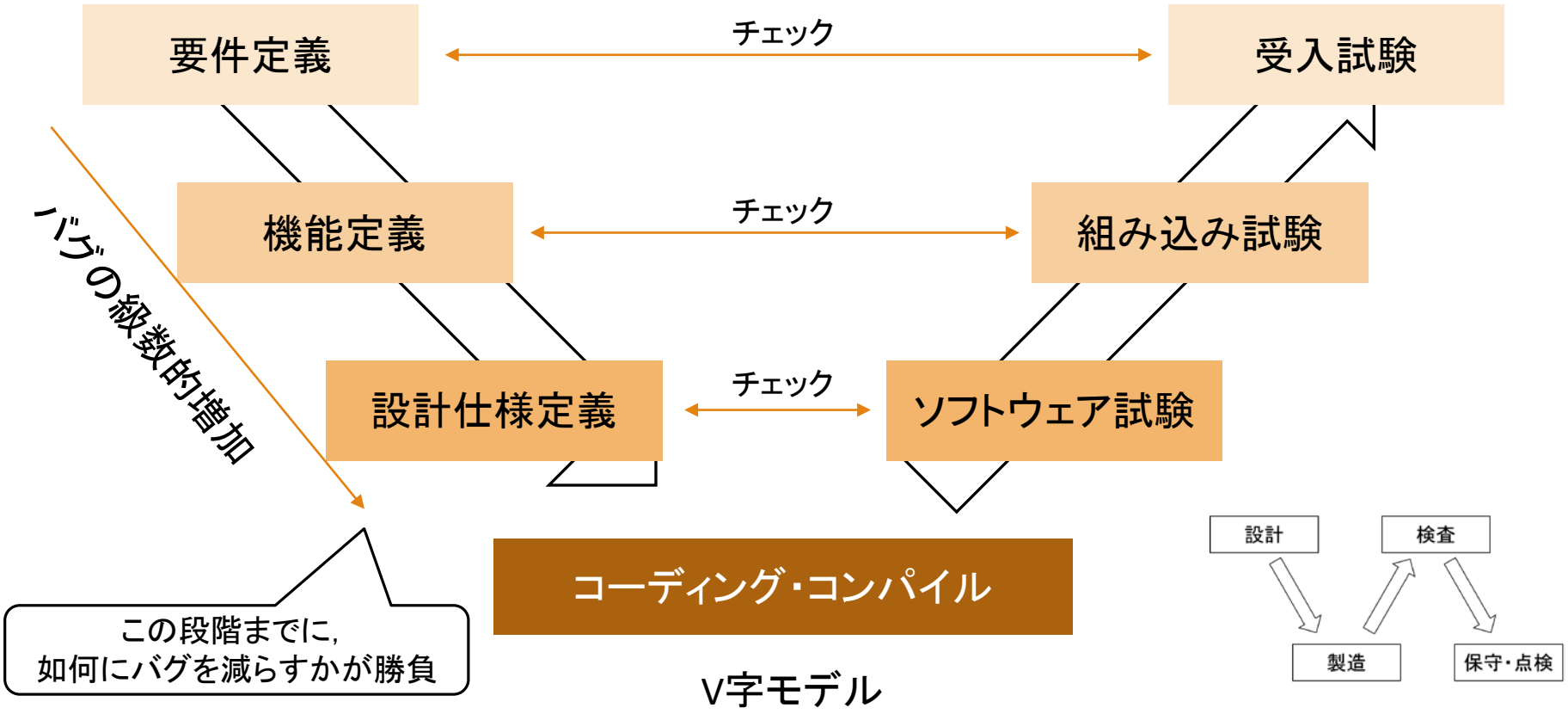
十中八九
プログラムに関する定義不足

レスバグ・プログラミング

3つのやるべきこと

- 要件定義・機能定義・設計仕様定義を紙に書く
- 粗くてもいいのでフローチャートを必ず書く
- 粗くてもいいので変数相関図を必ず書く

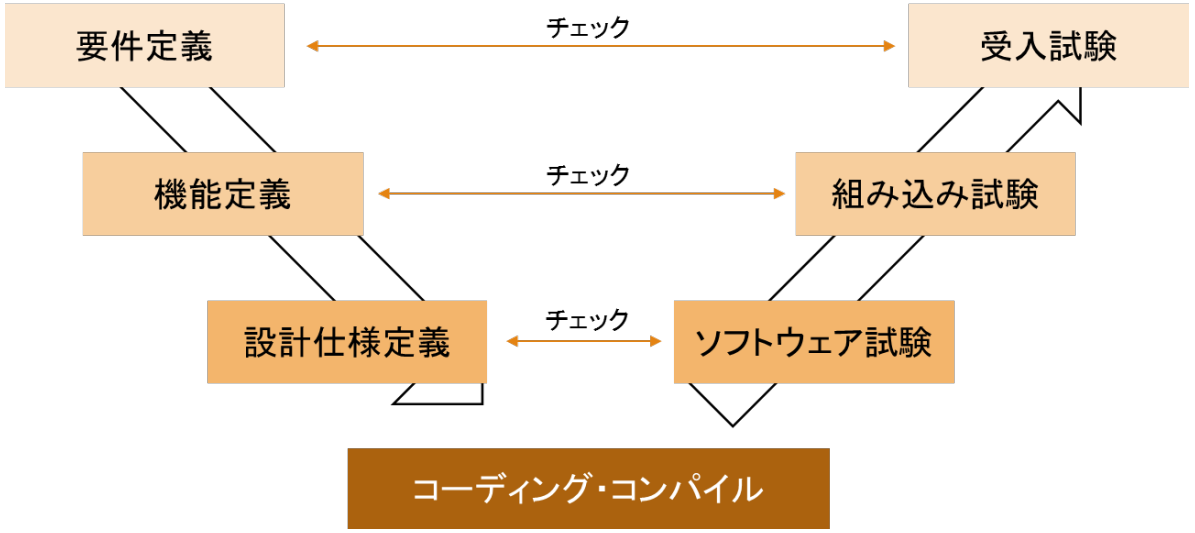
プログラミングの流れ



V字モデル

レスバグ・プログラミング

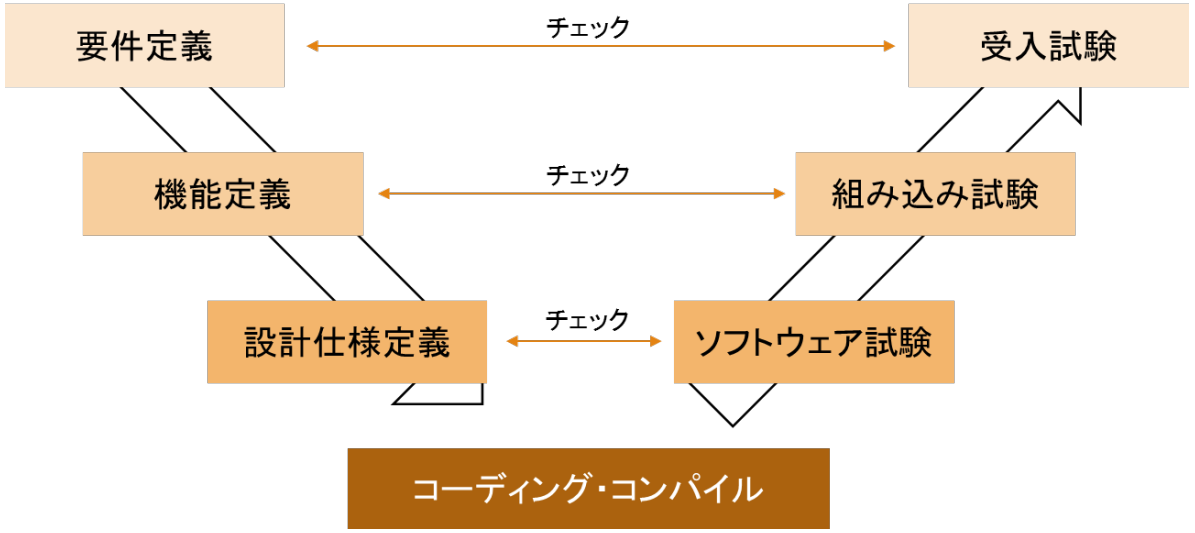
要件定義・機能定義・設計仕様定義を紙に書く
- 要件定義：何をどうしたいか（概念設計）
  - 例：入力した2つの行列の積を計算したい



レスバグ・プログラミング

要件定義・機能定義・設計仕様定義を紙に書く

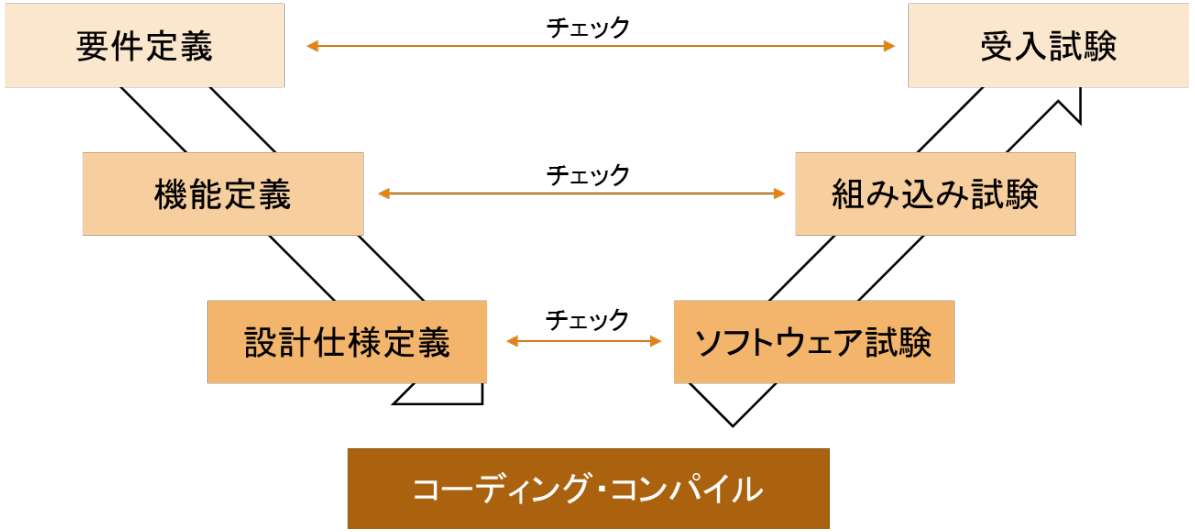
- 機能定義：どういう仕組みで要件定義を解決するか
  - 例：行列入力を受け取る機能＋積を計算できるかを考える機能＋積を出力する機能＋エラー処理＋…



レスバグ・プログラミング

要件定義・機能定義・設計仕様定義を紙に書く

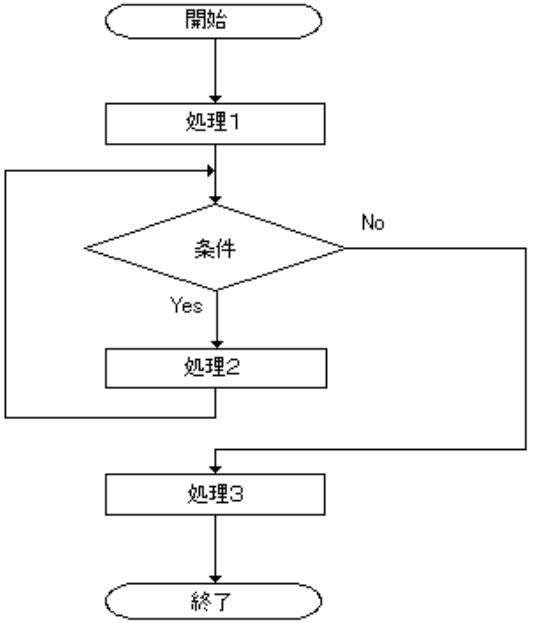
- 設計仕様定義：具体的にどのような手法で要件定義を解決するか
  - 例：関数・変数は何を使うか　クラスはどうするか　など



レスバグ・プログラミング

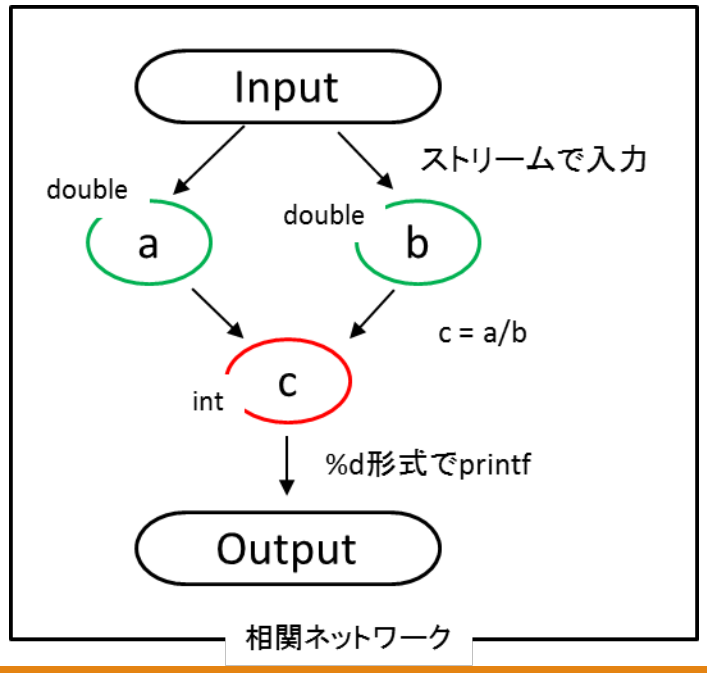
粗くてもいいのでフローチャートを必ず書く

- ここまでで最低でも機能定義が網羅できる
- 機能定義時点までのバグを発見可
- 慣れるまでは頭のなかでやらない



レスバグ・プログラミング

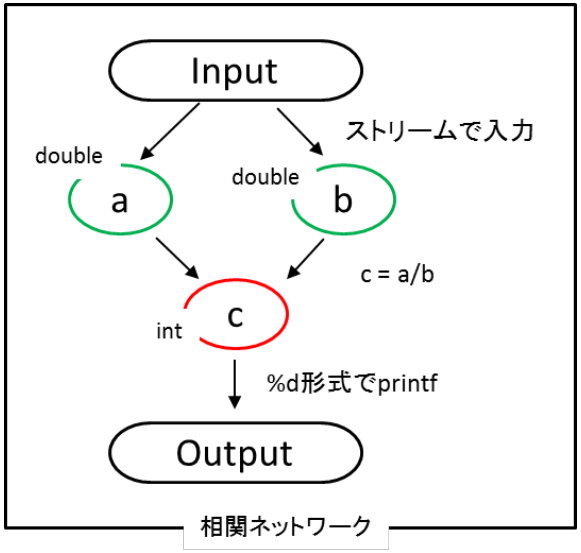
粗くてもいいので変数相関図を必ず書く
- ソフトウェア試験や，後のバグフィックスに役立つ
- 慣れるまでは頭のなかでやらない



変数相関図について

変数相関図（相関ネットワーク）とは？

- 入力・出力と全変数をノードとし，各ノードを注釈付きの矢印で結んだ図
  - 矢印につける注釈は，各ノード間の関係を記載する



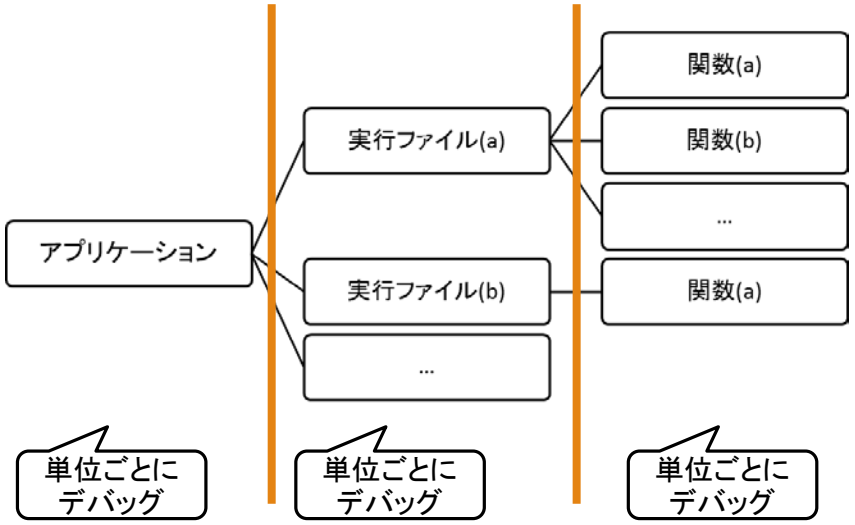
バグの発生箇所を特定

発生箇所の特定

プログラムの圧縮

- 全部をデバッグするのではなく，機能ごとにデバッグ

  例) 関数ごとにデバッグ



発生箇所の特定

兎に角ログを吐き出す

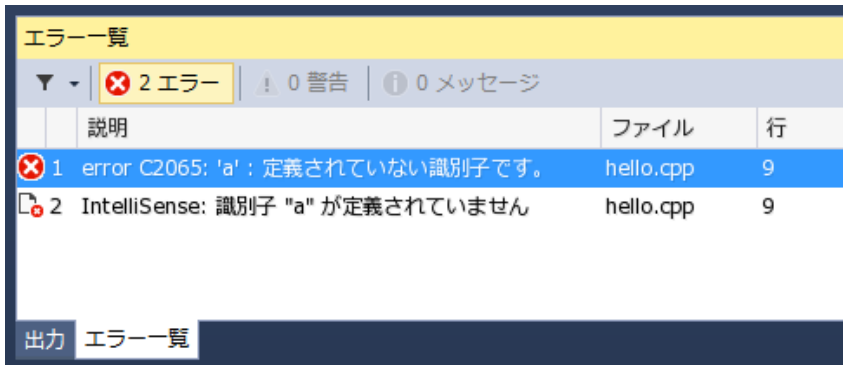
- 吐き出し方 :
  - printf, cout, 規定のテキストログ, ダンプファイル
- どこまで規定の動作をしているのかを見る
  - 規定の動作 = 仕様定義された動作
  - ここで規定の動作をいい加減に定めると，さらなるエラーを引き起こす

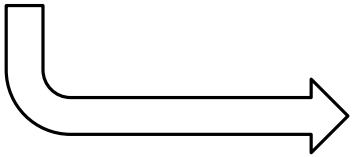
バグの原因を特定

原因の特定

表示されるエラーメッセージをよく読む

- 何言ってるかわからない場合は，変数部分を空白にした文言でググる





原因の特定

C初心者のやりがちなミス

- 未定義領域にアクセスをする（アクセス禁止法違反）

  例)a[100]の配列を生成→a[100]にアクセス
  int aを定義して，初期化せず cout するなど
- 誤った改行や文字(; など)の欠損
- 二重定義

  二重定義とは：同じ文字の変数を2回以上定義すること
  （リキャストと違う）

原因の特定

C初心者のやりがちなミス

- カウンタなどのオーバーフロー
- 代入ミス
  - 特にファイル場所の指定に多い．文字結合のミスとか
- 複数の main 関数の有るcファイルの混用
- 定義済みだと勘違いのマクロを使用
  - 特にM_PI

原因の特定

表示されるログをよく読む

- おかしいと思う部分以外を削除して(=最低限の環境で) コンパイル&実行

- それでもうまくいかない場合は,
  頭を空っぽにして，ソース通りに手計算
  - 暗算はしない．必ず紙に書く
  - 当然省略しない．全部を紙に書き出す．

バグの駆逐

バグの駆逐で怖いこと

修正が生むさらなるバグ

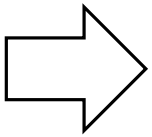
例) 何がよくないか？

- 途中で出力が0になっており，原因を探したらdouble型を含む計算をint型で行っていた．そのため，int型定義でなく変数定義を全てdouble型で行った．

```
#include <iostream>
#include <stdio.h>

int main(void)
{
    double a;
    double b;
    std::cin >> a >> b;
    int c;
    c = a / b;
    printf("%d¥n", c);
    system("pause");
}
```

```
2.0 4.0
0
続行するには何かキーを押してください . . .
```



```
#include <iostream>
#include <stdio.h>

int main(void)
{
    double a;
    double b;
    std::cin >> a >> b;
    double c;
    c = a / b;
    printf("%d¥n", c);
    system("pause");
}
```

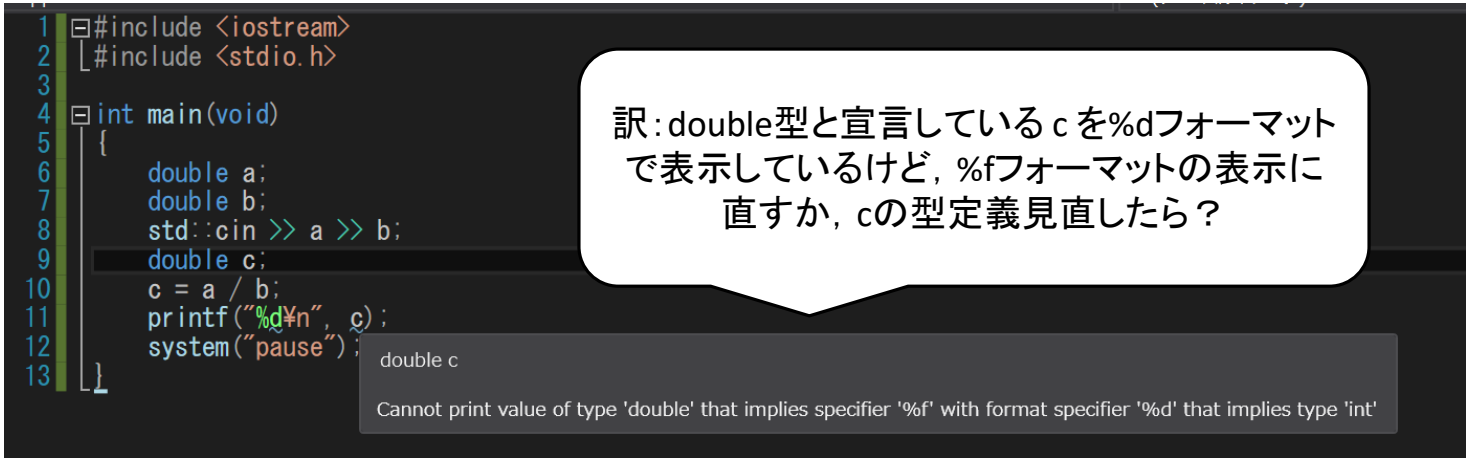
バグの駆逐で怖いこと

修正が生むさらなるバグ

例) 何がよくないか？

- 途中で出力が0になっており，原因を探したらdouble型を含む計算をint型で行っていた．そのため，int型定義でなく変数定義を全てdouble型で行った．

printfの表記が直っていないので×



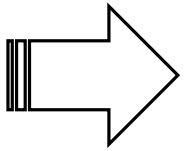
バグの駆逐で怖いこと

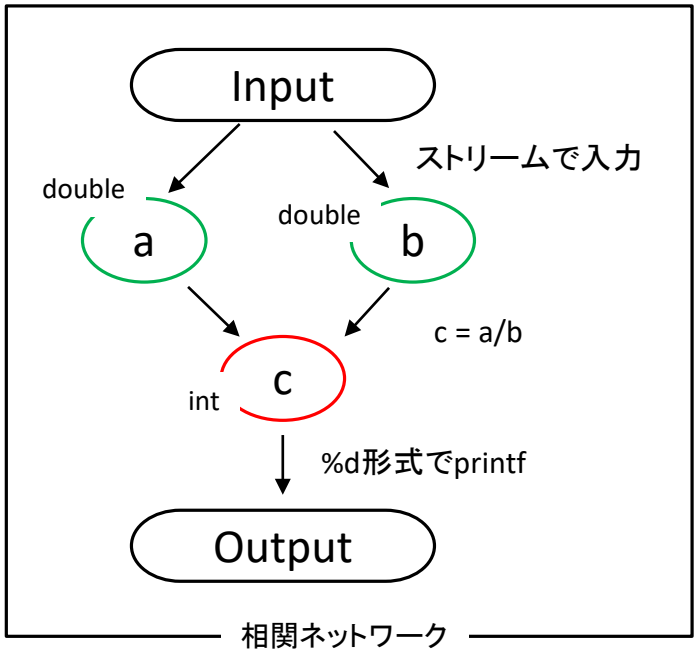
修正が生むさらなるバグ

- 修正する変数のステークホルダも必ず修正必要か確認

```
#include <iostream>
#include <stdio.h>

int main(void)
{
    double a;
    double b;
    std::cin >> a >> b;
    int c;
    c = a / b;
    printf("%d¥n", c);
    system("pause");
}
```

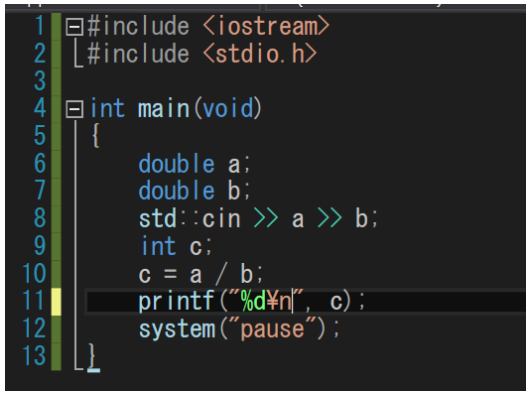




バグの駆逐で怖いこと

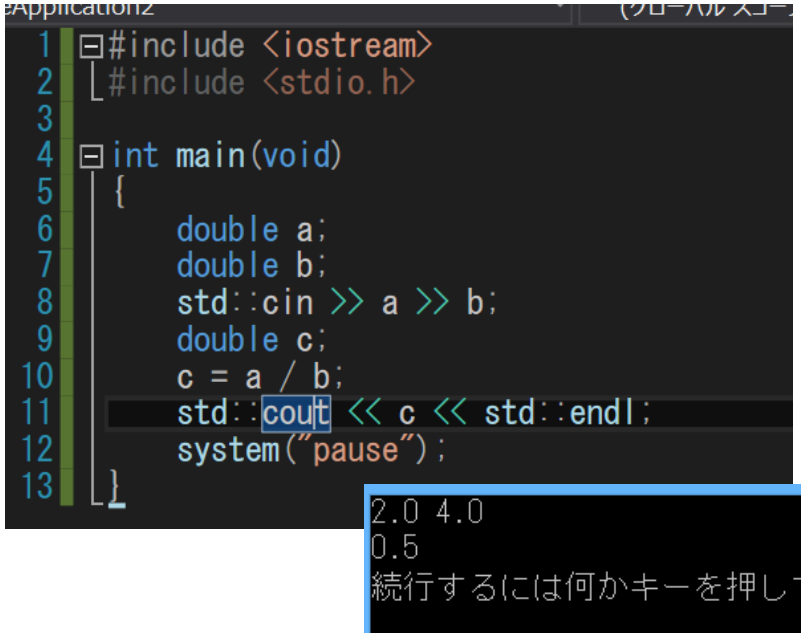
修正が生むさらなるバグ

- 修正後はバグの起こりにくい形に変える



```
#include <iostream>
#include <stdio.h>

int main(void)
{
    double a;
    double b;
    std::cin >> a >> b;
    int c;
    c = a / b;
    printf("%d¥n", c);
    system("pause");
}
```



```
#include <iostream>
#include <stdio.h>

int main(void)
{
    double a;
    double b;
    std::cin >> a >> b;
    double c;
    c = a / b;
    std::cout << c << std::endl;
    system("pause");
}
```

```
2.0 4.0
0.5
続行するには何かキーを押してください . . .
```

修正を試験する

修正を試験する

バグを分類する

- アウトプット・マトリクス

あるべき		Output	
		Ture	False
Input	True	○	-
	false	-	○

おかしい出力

例		Output	
		Ture	False
Input	True	○	○
	false	-	○

1. どのようなinputをしたらどのようなoutputがあるかを書き下す
2. あるべきアウトプットマトリクスとの差を見る

試験の注意

評価シートの作成

- 出来る限り多くの種類の観点で試験する
  - アウトプットマトリクスを意識する

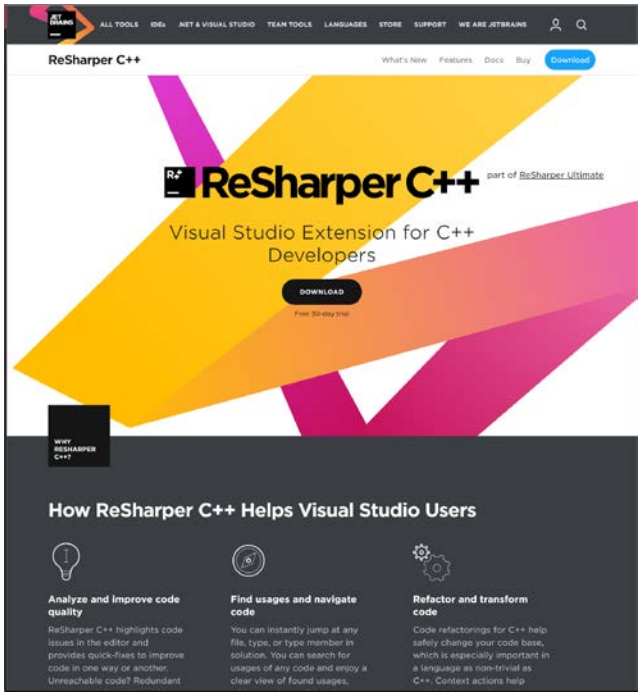
例) .txt ファイルの読み込み部分の試験

ID	内容	結果
000	.txtを入力	
001	.rawを入力	
002	.csvを入力	
003	入力に何も指定しない	
004	拡張子を変更した元.txtを入力	
005	拡張子を変更した元.rawを入力	
…	…	

Appendix

JETBRAINS ReShaper C++

- 言語は英語限定（日本語環境でも使用可能）
  - 怖い人は英語環境での使用を推奨
- 学生版無料
- 定義不足や未定義のところを赤字にして
  バグを防止してくれる
- C#だと更にリッチな機能（変数の自動修正）が
  ある



* JETBRAINSはPythonのエディターで有名
(Jetbrains PyCharm Community)