情報処理実習
第8回: 自作関数

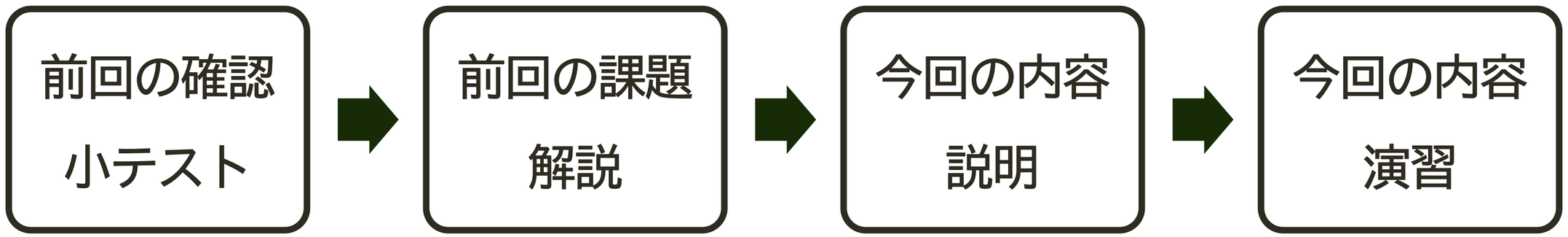
担当: 佐藤

2018年11月12日(月)

アナウンス

- 授業前にやること
  - 計算機にログオン
  - CoursePowerにログイン
- 注意事項
  - 教室内飲食禁止
  - 原則として，授業中ではなく休み時間にトイレにいきましょう

授業の進め方



基本事項の確認小テスト

制限時間厳守

① CoursePowerにログインして，すぐに解答が始められるよう準備する

② 開始の合図があるまで，解答を始めずに待機する

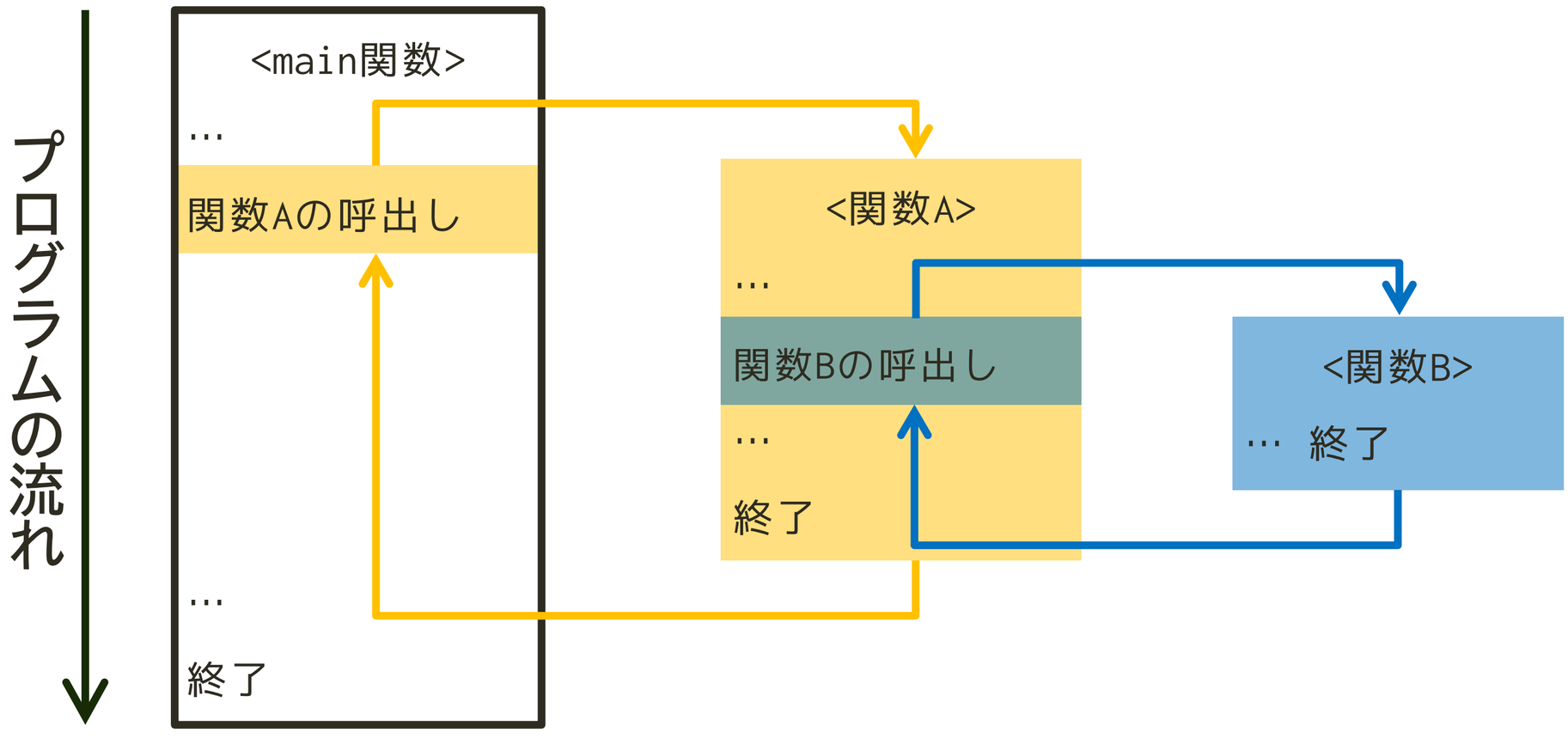
注意

「閉じる」ボタンは決して押さないこと!

関数の概要

C言語の関数

- C言語の**関数**は，プログラムの「部品」
  - 関数を呼び出すと，プログラムの流れが呼び出した関数へ移り，処理が終わると呼び出し元に戻る



自作関数の作り方

- C言語では，あらかじめ用意された関数(例：main()，printf()，scanf()など)とは別に，自作の新しい関数(自作関数)を作成できる
  - 自作関数を作成することを**関数定義**という(※1)

基本的に好きな
名前をつけられる(※2)

int，doubleなど

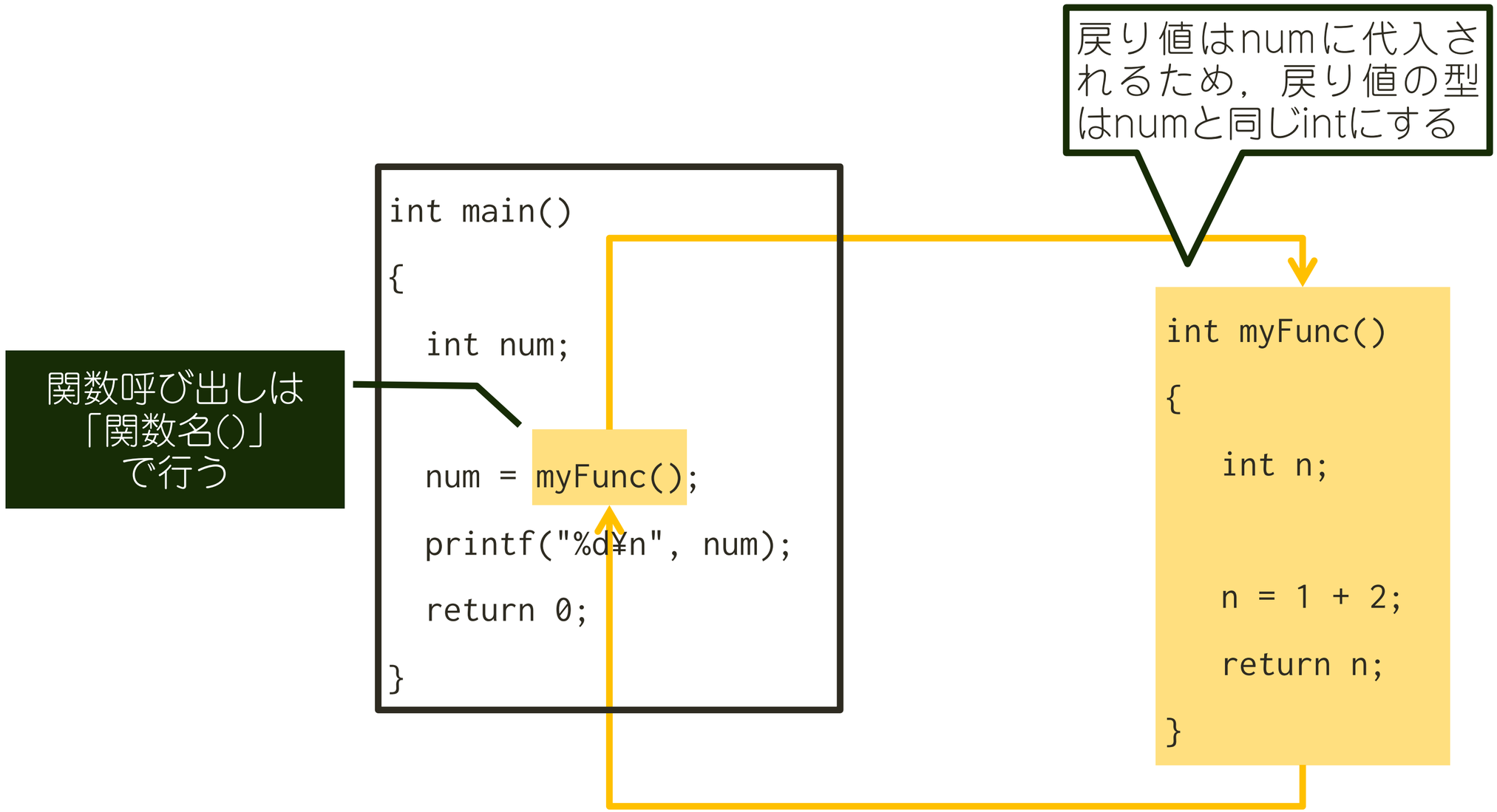
文法

```
戻り値の型 関数名()
{
    …
    return 戻り値;
}
```

呼び出し元に返す値を書く．
変数や計算式でもよい

※1 C言語では，関数定義はネストしない(Visual Studio 2015では，ビルドエラー)
※2 厳密には「先頭は数字であってはならない」ことなど，変数名と同じ制約がある

関数の使い方



- main()のmyFunc()の部分が3になり，numに代入される

自作関数を作る場面

① 何度も使う記述を1つにまとめる
- あるまとまった記述を関数にしておくと，main()が簡潔になるためどんな内容のプログラムなのかがわかりやすくなる

② プログラムを機能別にまとめる
- プログラムの機能修正が生じた場合，その機能に対応する関数だけを修正すればよくなる

①何度も使う記述を1つにまとめる

```
int main()
{
  …
  printf("Game Over");
  …
  printf("Game Over");
  …
  printf("Game Over");
  …
  return 0;
}
```

```
int main()
{
  …
  showGameOver();
  …
  showGameOver();
  …
  showGameOver();
  …
  return 0;
}
```

戻り値がない場合，
戻り値の型はvoidにする

```
void showGameOver() {
  printf("Game Over");
}
```

戻り値がない場合，
return文は省略可

- 表示する文字が「Game Over」から変更された場合，手作業でひとつひとつ直すのは大変!関数にしておけば，1箇所(関数の中身)を変更するだけで済む

引数を伴う自作関数の作り方

- 関数を呼び出す際，関数に値を渡すことができる．ただし，関数にあらかじめ**引数(ひきすう)リスト**を書いておく必要がある

文法

```
戻り値の型  関数名(引数リスト)
{
    …
    return 戻り値;
}
```

引数が3つの例:
int a, int b, double c

- **引数**とは，呼び出し元と関数の間で値の受け渡しを行うための変数のこと
  - 関数側の引数は，データ型を伴って「**データ型 引数名**」と書く
  - 呼び出し側の引数にはデータ型を書かない
- 引数リストは，個々の引数を**「,」で区切って**並べたもの
- 引数は，受け渡しする値の数だけ用意する

②プログラムを機能別にまとめる

```
int main()
{
  int num;

  num = 1 + 2;

  printf("%d", num);

  return 0;
}
```

```
int main()
{
 int num;

  num = sum(1, 2);

  printf("%d", num);

  return 0;
}
```

1と2がそれぞれ実引数

実引数はデータ型を伴わない

```
int sum(int a, int b)
{
  return a + b;
}
```

aとbがそれぞれ仮引数

仮引数はデータ型を伴う

- 関数の引数リストにあらかじめ書いておく個々の引数を仮引数(かりひきすう)，呼び出し元から関数に渡す実際の値を実引数(じつひきすう)という
- **実引数と仮引数の順序は一致**する．上の例では，1がaに，2がbに渡される
- 実引数と仮引数のデータ型を一致させること

注意

- 仮引数と実引数の名前が同じであっても，それぞれ別の変数として扱われる
  - もちろん，別々の名前にしてもよい

```c
#include <stdio.h>

void swap(int a, int b) {
  int tmp = a;

  a = b;
  b = tmp;
}

(右へ続く)
```

aとa，bとbは，
それぞれ別の変数

```c
int main()
{
  int a, b;

  scanf("%d", &a);
  scanf("%d", &b);

  swap(a, b);

  printf("%d, ", a);
  printf("%d¥n", b);

  return 0;
}
```

自作関数を書く場所は
main()の前

ライブラリ

ライブラリ

- C言語では，あらかじめ用意されている便利な関数を利用してプログラムを作成できる．このあらかじめ用意されている関数の集まりをライブラリという
- 機能ごとに分けられたいくつかのライブラリが用意されており，**プログラムの冒頭で**特定のライブラリに対応するヘッダファイル(～.h)をインクルード(#include <～.h>)することで，そのライブラリに含まれる関数が使用可能になる
  - 例: 標準入出力ライブラリに含まれる入出力関数を利用したい
    - プログラムの冒頭で「#include <stdio.h>」と書いておく
    - printf()，scanf()などが利用可能になる
  - 例: 数学ライブラリに含まれる数学関数を利用したい
    - プログラムの冒頭で「#include <math.h>」と書いておく
    - fmax()，fmin()，fabs()などが利用可能になる

よく使われるライブラリ

ライブラリの名前	対応するヘッダファイルの名前	ライブラリに含まれる関数の機能
標準入出力ライブラリ	stdio.h （スタンダード・アイオー）	基本的な入出力を可能にする
数学ライブラリ	math.h （マス）	基本的な数学の演算を可能にする
文字列操作ライブラリ	string.h （ストリング）	基本的な文字列操作を可能にする

変数の有効範囲

グローバル変数とローカル変数

- 関数の**外部**で宣言された変数のことを**グローバル変数**という
  - グローバル変数は，どの関数からも共通して利用可能
  - 複数の関数の中で値を更新する必要がある変数は，グローバル変数として宣言しておくとよい
  - グローバル変数を参照するどの関数よりも前で宣言しておく必要がある
    - 通常，**すべての関数の前(プログラム先頭部分)に書く**
- 関数の**内部**で宣言された変数のことを**ローカル変数**という
  - ローカル変数は，**宣言された関数の中だけで使える**
  - その関数の中でしか使われない変数は，原則としてグローバル変数ではなくローカル変数として宣言しておくこと
- このように，変数には**有効範囲(スコープ)**がある

グローバル変数の使用例

初期値を設定しない場合，グローバル変数は
必ず0で初期化される(ここでは，num == 0)

```c
#include <stdio.h>

int num; /* グローバル変数 */

void incrementNum(int x) {
    num = num + x;
}

void decrementNum(int x) {
    num = num - x;
}
```

(右へ続く)

```c
int main() {
    printf("%d¥n", num);
    incrementNum(1);
    printf("%d¥n", num);
    decrementNum(1);
    printf("%d¥n", num);

    return 0;
}
```

異なる関数の中でグローバル変数は
宣言しなくても使用可能

グローバル変数の注意点

- **変数のスコープは最小範囲にすることが望ましい**
  - グローバル変数は複数の関数内で使用可能➔グローバル変数に関わる不具合が生じた場合，問題箇所の特定が困難
    - グローバル変数を使ったプログラムは，ローカル変数の受け渡しを行うことでグローバル変数を使わないプログラムに書き換えられる➔ローカル変数で事足りる場合，グローバル変数ではなくローカル変数を使うこと!

```
#include <stdio.h>

int incrementNum(int num, int x) {
    return num + x;
}

int decrementNum(int num, int x) {
    return num - x;
}

(右へ続く)
```

```
int main() {
    int num = 0;

    printf("%d¥n", num);
    num = incrementNum(num, 1);
    printf("%d¥n", num);
    num = decrementNum(num, 1);
    printf("%d¥n", num);

    return 0;
}
```

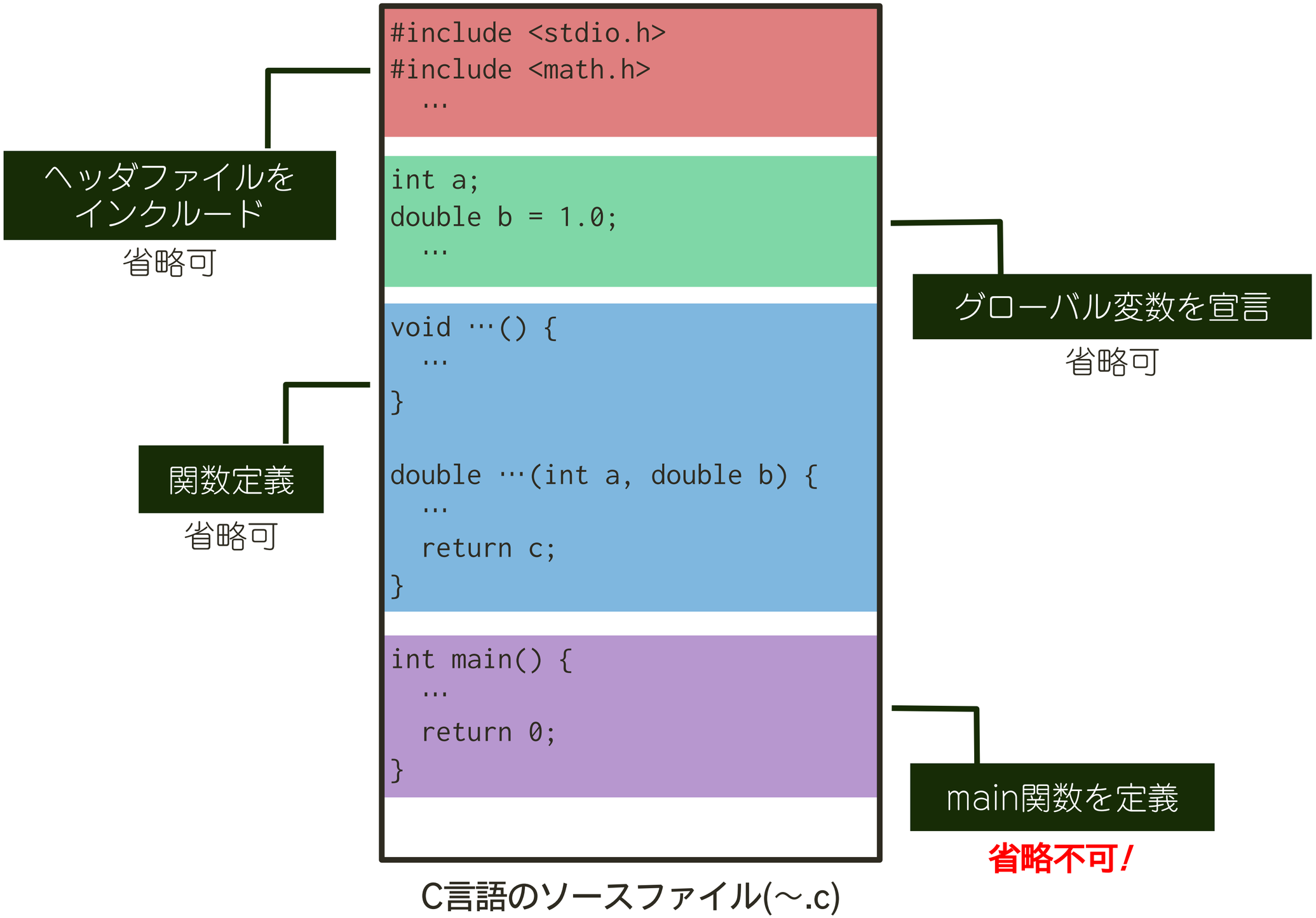
グローバル変数に関わる不具合の例
変数スコープの優先度

- スコープが異なる➔同じ名前の変数を宣言可能
  - **最も内側の(適用範囲の狭い)スコープを持つ変数として解釈される**
    - 同じ名前を持つ変数をグローバル変数とローカル変数として別々に宣言した場合，ローカル変数が優先される

```
#include <stdio.h>

int num; /* グローバル変数 */

int main() {
   int num = 1; /* (main()の)ローカル変数 */

   printf("%d¥n", num);

   return 0;
}
```

C言語のソースファイルの構造

これまでに学習した範囲内



課題

レポートの作成

① レポートの冒頭に以下を適切なレイアウトで書く
- 「情報処理実習第8回課題レポート」というタイトル
- 学生番号
- 氏名

② 課題ごとに以下を載せる
- 作成したプログラムのソースコード
- 作成したプログラムの実行結果を示すコマンドプロンプトのスクリーンショット

③ 完成したレポートをCoursePower上で提出する

提出〆切: 18:30