情報処理実習

第10回：自作関数

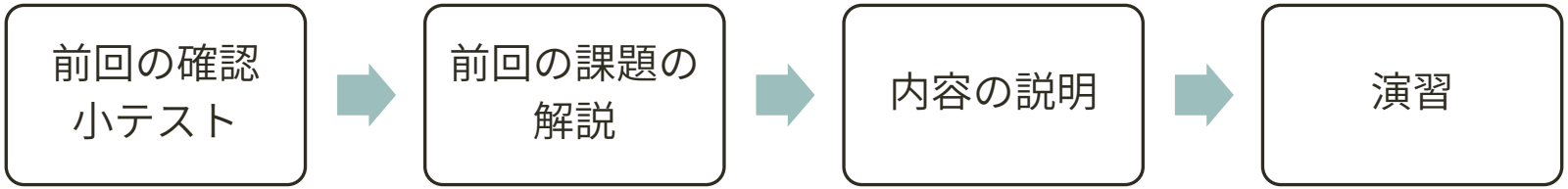
2017年11月27日(月)

担当：佐藤

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(確認)



基本事項の確認小テスト

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！

前回課題の解説

課題9-1

```
#include <stdio.h>

int main() {
    int number;
    int price[5] = { 200, 450, 300, 250, 500 };

    printf("1〜5番のうち、値段を知りたい商品の番号を入力して下さい。¥n");
    scanf("%d", &number);
    printf("%d番の商品の値段は%4d円です。¥n", number, price[number - 1]);

    return 0;
}
```

配列の初期化における初期化子は，個々の配列要素の初期化子を「,」で区切って並べたものを「{}」で囲む

配列の要素番号は0から始まるため，1から数えてn番目は，配列のn-1番目の要素

課題9-2

```
#include <stdio.h>

int main() {
   int i;
   double avg;
   double total = 0;

   double weight[10] = { 60.5, 72.3, 69.4, 55.5, 93.8,
                         48.0, 61.1, 67.8, 58.9, 63.6 };
   for (i = 0; i < 10; i++) {
      total += weight[i];
   }
   avg = total / 10;
   for (int i = 0; i < 10; i++) {
      printf("%2d番目の人の体重%5.1fkg：平均との差：%5.1fkg¥n", i + 1, weight[i], weight[i] - avg);
   }
   printf("平均体重：%5.1fkg¥n", avg);

   return 0;
}
```

10人の体重の平均を求める

番号調整

課題9-3(1)

配列要素の初期値の設定に初期化を用いる場合の解答例

2次元配列の初期化．左の[]の要素数は外側の{}の初期化子の数に対応する．右の[]の要素数は内側の{}の初期化子の数に対応する

```
#include <stdio.h>

int main() {
   int i, point1;
   int score[3][3] = { { 0, 1, 2 }, { 2, 3, 4 }, { 4, 5, 6 } };

   printf("1回目の結果は？¥n");
   scanf("%d", &point1);
   for (i = 0; i < 3; i++) {
      printf("2回目の結果：%d, 合計得点：%d", i, score[i][point1]);
      printf("¥n");
   }

   return 0;
}
```

外側の配列の要素番号が2回目の得点，内側の配列の要素番号が1回目の得点，内側の配列の要素の値が合計得点に対応している．例えば，1回目の得点が2，2回目の得点が1の場合の合計得点は，外側の1番目の配列の2番目の要素の値，すなわち4となる

課題9-3(2)

配列要素の初期値の設定に代入を用いる場合の解答例

```
#include <stdio.h>

int main() {
   int i, j, point1, score[3][3];

   for (i = 0; i < 3; i++) {
      for (j = 0; j < 3; j++) {
         score[i][j] = (i * 2) + j;
      }
   }
   printf("1回目の結果は？¥n");
   scanf("%d", &point1);
   for (i = 0; i < 3; i++) {
      printf("2回目の結果：%d, 合計得点：%d", i, score[i][point1]);
      printf("¥n");
   }

   return 0;
}
```

配列要素に合計得点を設定

要素番号と合計得点の間の関係式

		1回目		
		白(0)	赤(1)	黄(2)
2回目	白(0)	0	1	2
	赤(1)	2	3	4
	黄(2)	4	5	6

+2 +2 +1 +1

課題9-4(1)

配列要素の初期値の設定に初期化を用いる場合の解答例

```
#include <stdio.h>

int main() {
   int i, j;
   int matrixC[3][4];
   int matrixA[3][4] = { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 } };
   int matrixB[3][4] = { { 1, 3, 5, 7 }, { 9, 11, 13, 15 }, { 17, 19, 21, 23 } };

   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         matrixC[i][j] = matrixA[i][j] + matrixB[i][j];
      }
   }
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         printf(" %2d", matrixC[i][j]);
      }
      printf("¥n");
   }

   return 0;
}
```

行列の和を格納するための配列

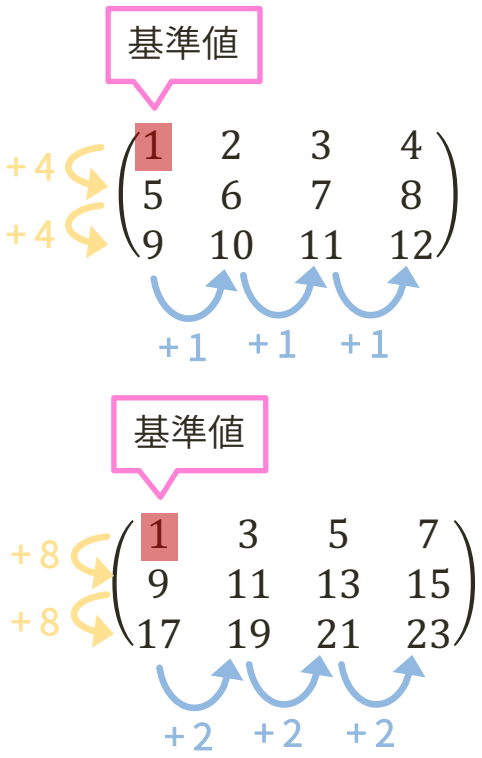
2重for文によって2次元配列のすべての要素にアクセス可能．外側のfor文は外側の配列の要素に順にアクセスするために用い，内側のfor文は内側の配列の要素に順にアクセスするために用いる

%の前のスペースを消して%3dとしてもよい

課題9-4(2)

配列要素の初期値の設定に代入を用いる場合の解答例

```c
#include <stdio.h>

int main() {
   int i, j;
   int matrixA[3][4], matrixB[3][4], matrixC[3][4];

   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         matrixA[i][j] = (i * 4) + (j + 1);
      }
   }
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         matrixB[i][j] = (i * 8) + (j * 2 + 1);
      }
   }
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         matrixC[i][j] = matrixA[i][j] + matrixB[i][j];
      }
   }
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 4; j++) {
         printf(" %2d", matrixC[i][j]);
      }
      printf("¥n");
   }

   return 0;
}
```



課題9-4(3)

除法演算子(/)と剰余演算子(%)を活用した解答例

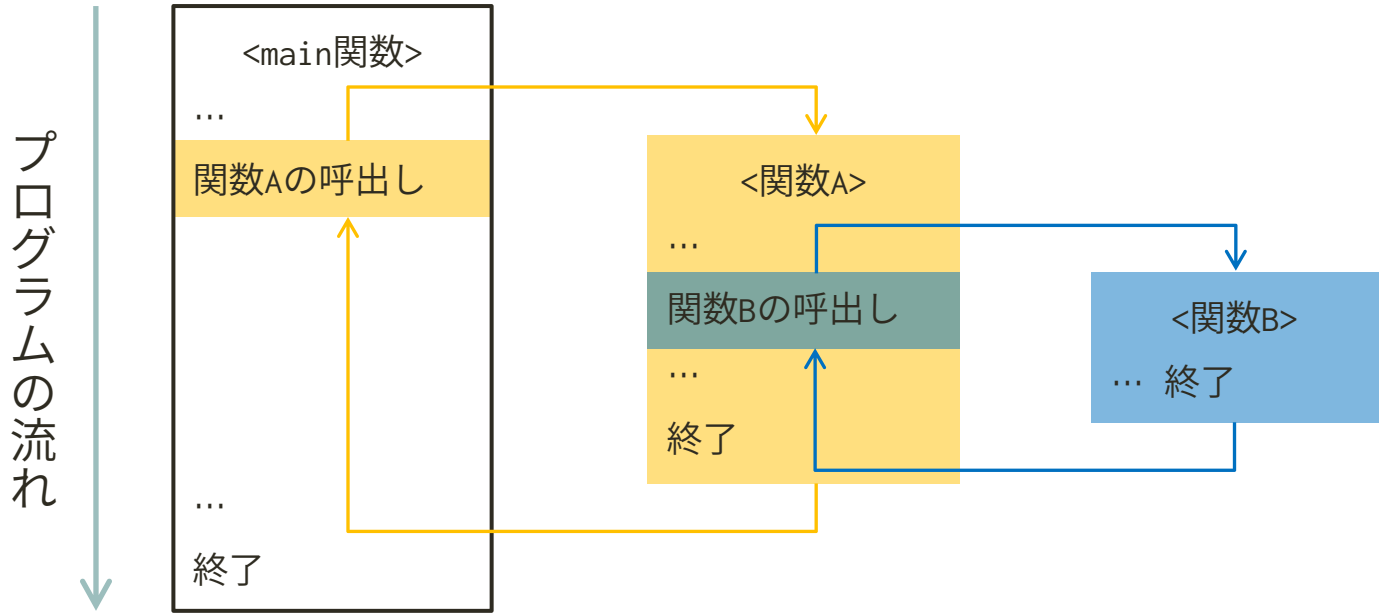
独力で解読してみよう！

```
#include <stdio.h>

int main() {
   int i, matrixA[3][4], matrixB[3][4], matrixC[3][4];

   for (i = 0; i < 12; i++) {
      matrixA[i / 4][i % 4] = i + 1;
      matrixB[i / 4][i % 4] = (i * 2) + 1;
      matrixC[i / 4][i % 4] = matrixA[i / 4][i % 4] + matrixB[i / 4][i % 4];
      printf("%3d", matrixC[i / 4][i % 4]);
      if (i % 4 == 3) printf("¥n");
   }

   return 0;
}
```

関数の概要

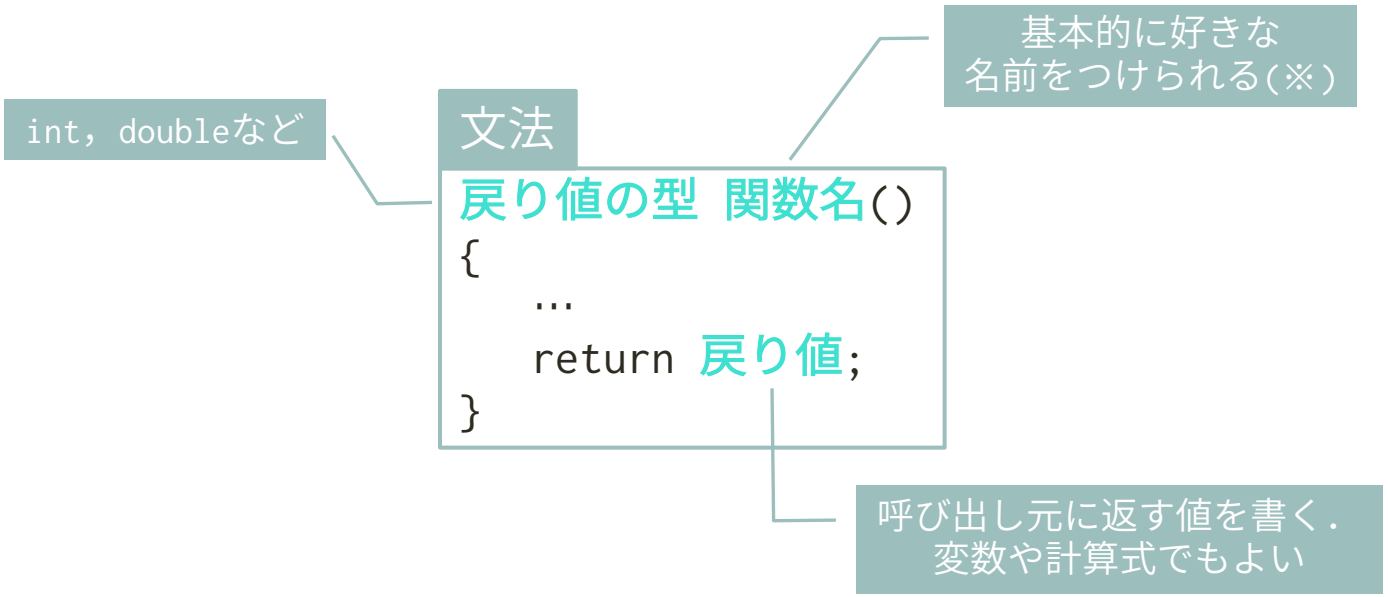
C言語の関数

- 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



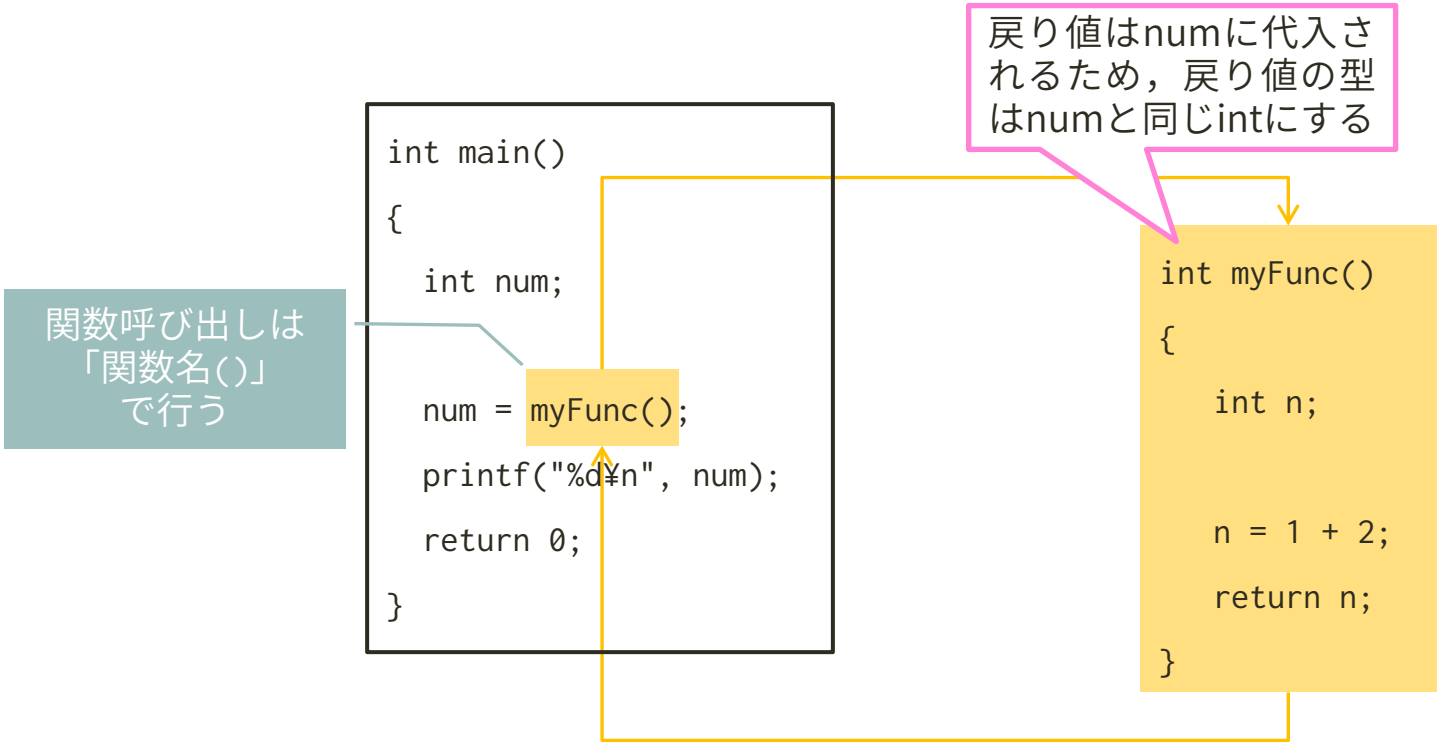
自作関数の作り方

- 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



※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



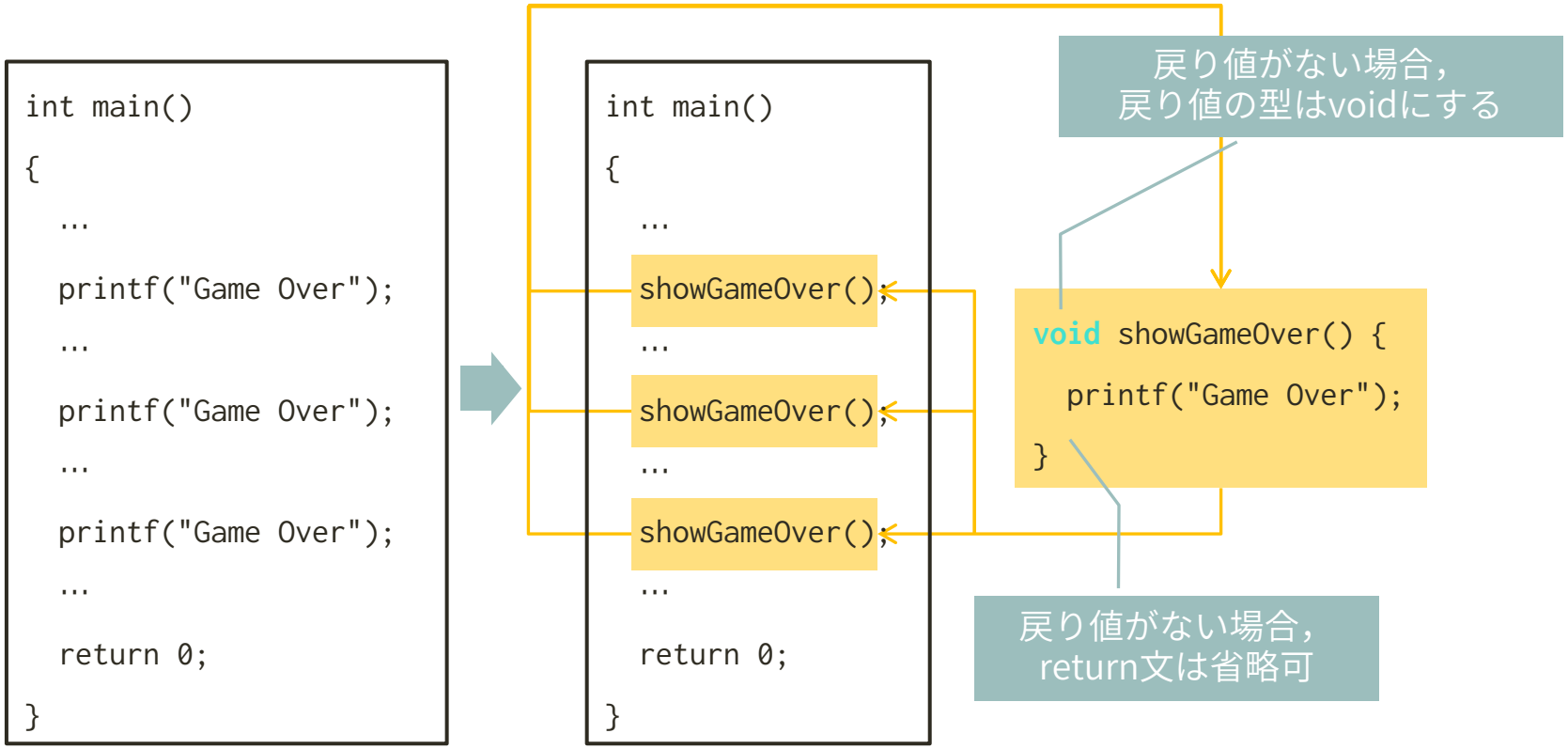
- main関数のmyFunc()の部分が3になり，numに代入される

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関数が簡潔になるため，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



- 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

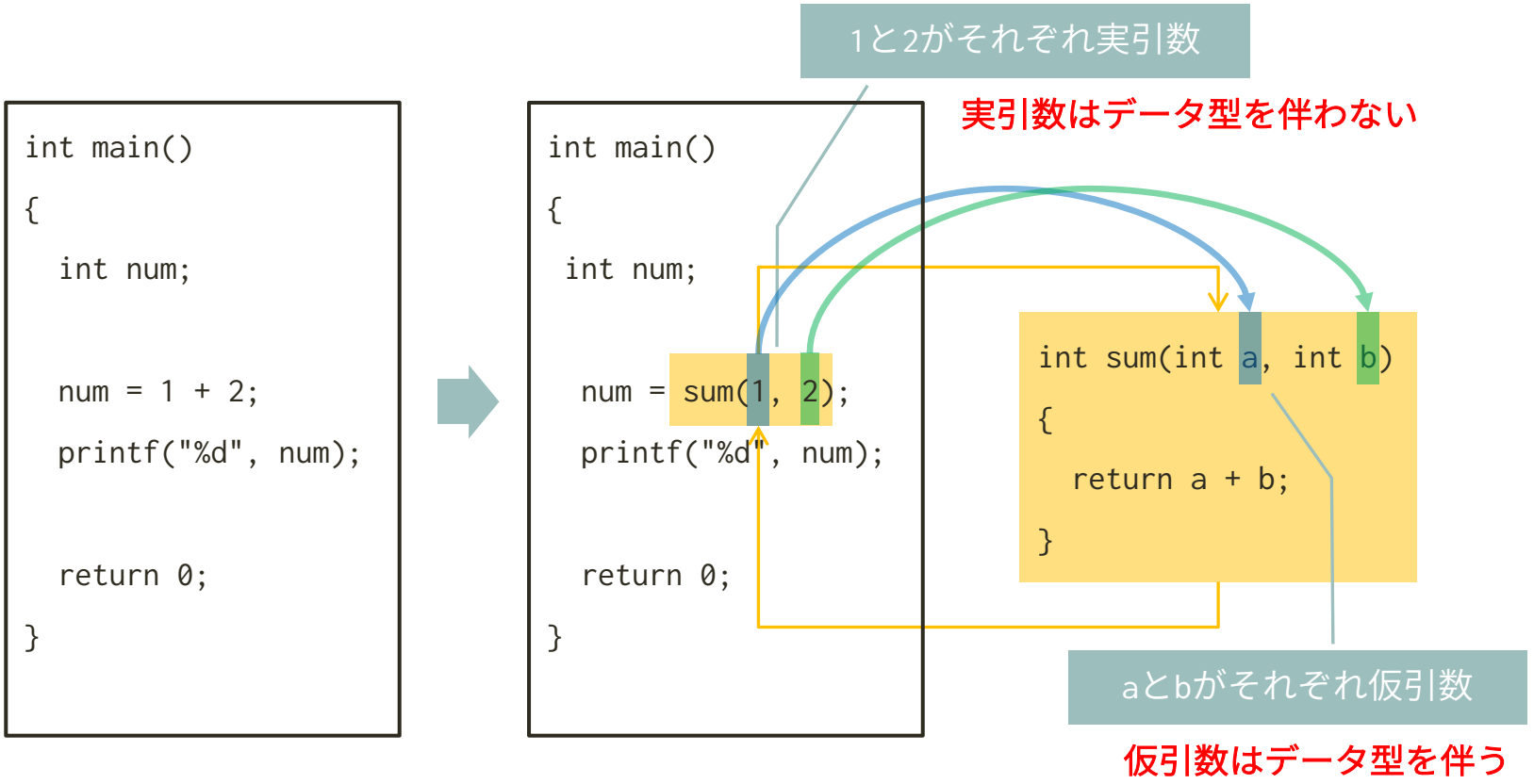
文法

```
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

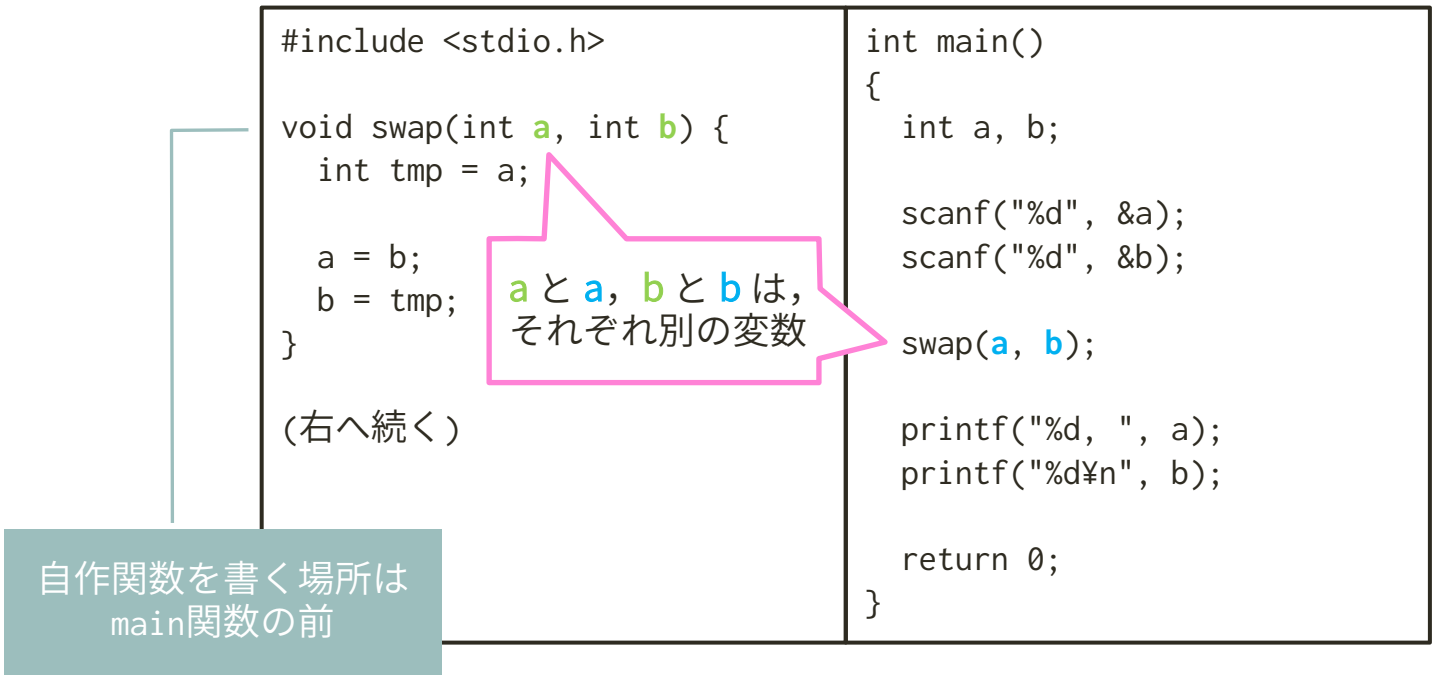
②プログラムを機能別にまとめる



- 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



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関数，scanf関数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関数，fmin関数，fabs関数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（スコープ）

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でしか使えない**
  - 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)

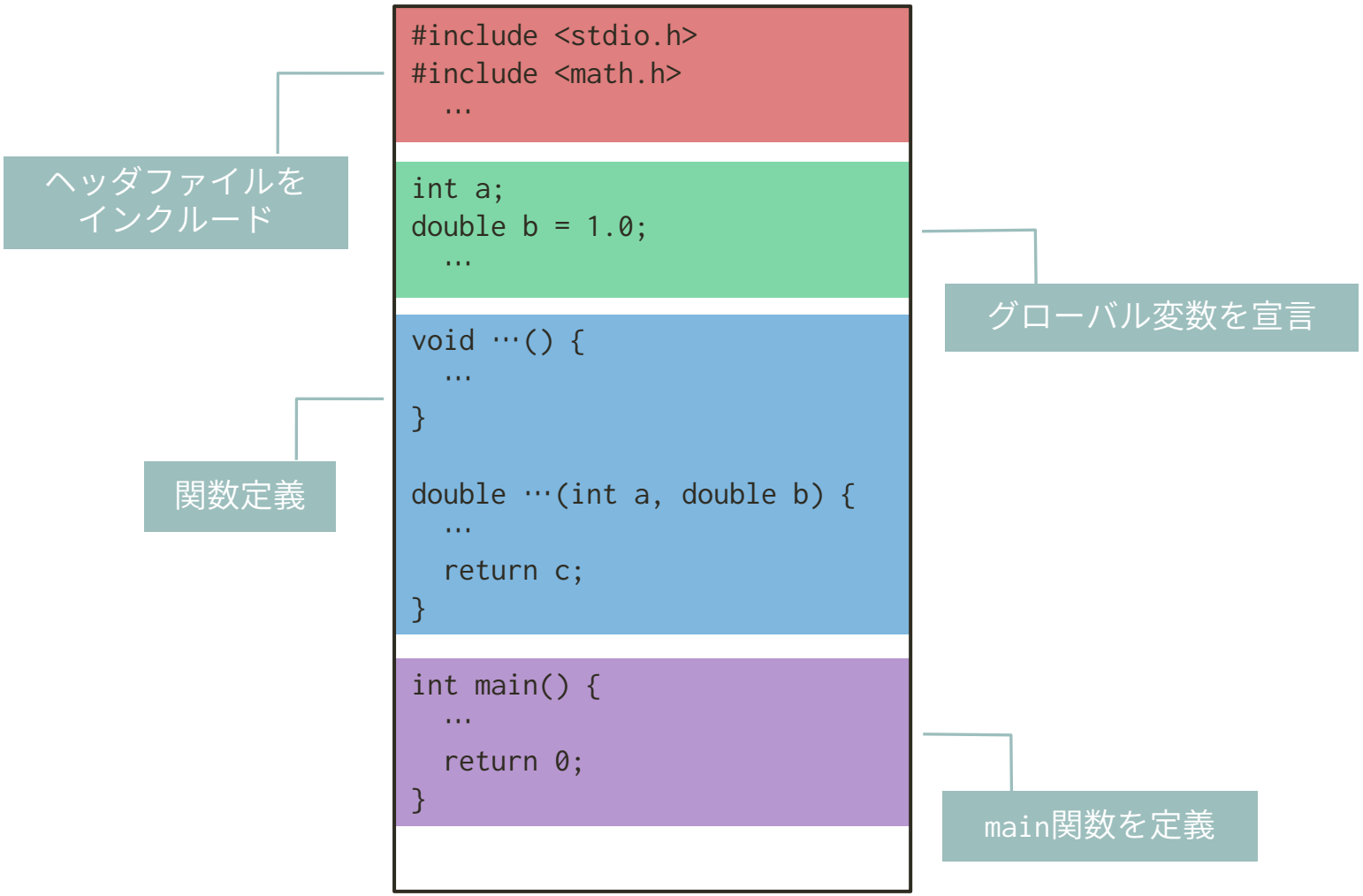
```c
#include <stdio.h>

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

C言語のソースファイルの構造



C言語のソースファイル(～.c)

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10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