cp-8. 関数

（C プログラミング入門）

URL: https://www.kkaneko.jp/cc/adp/index.html

金子邦彦

内容

例題１．棒グラフ

　　関数とは．プログラム実行順．データの流れ．

　　引数と仮引数

例題２．月の日数

　　return 文

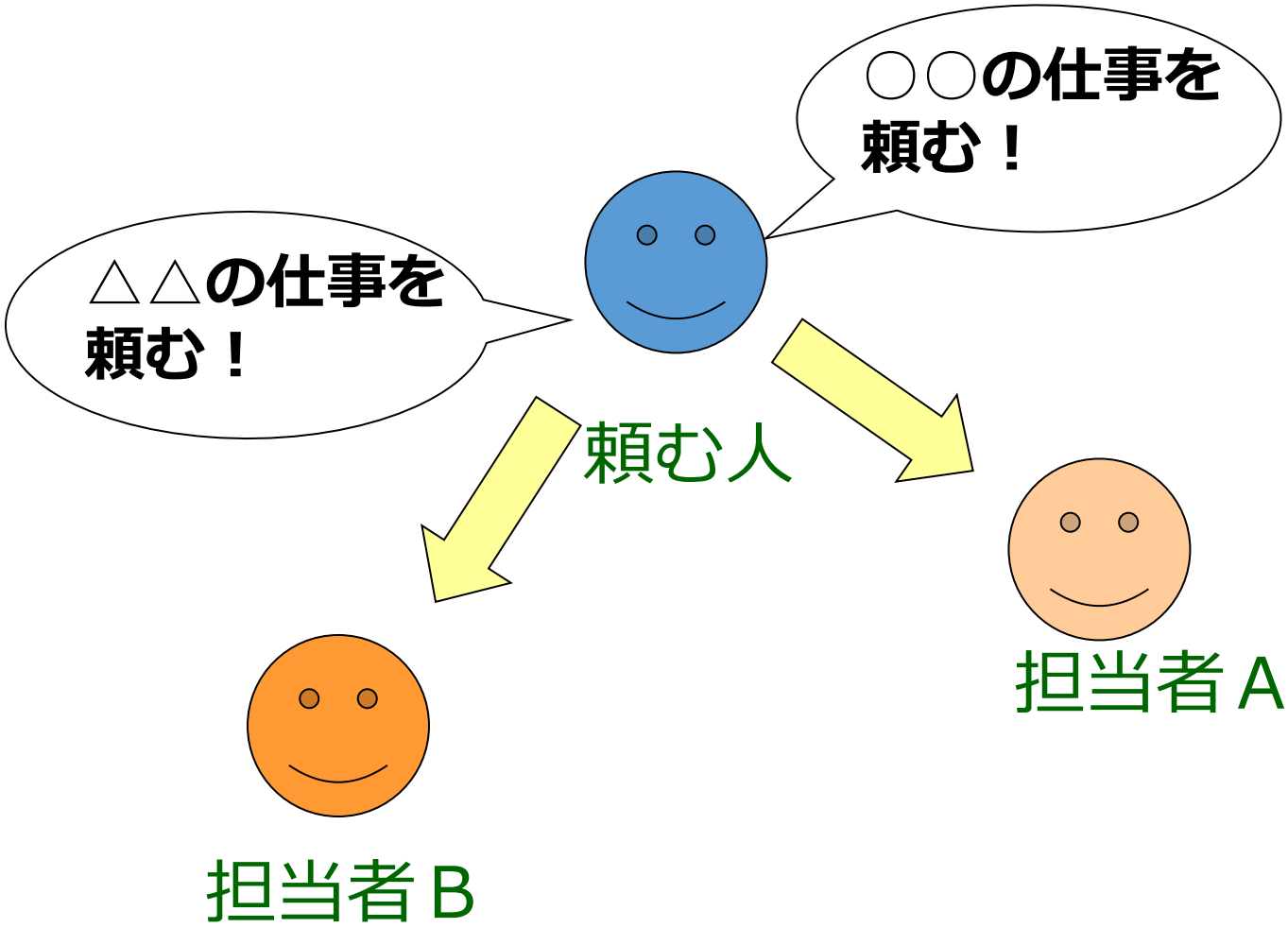
例題３．１か月分のカレンダー

例題４．月初めの曜日

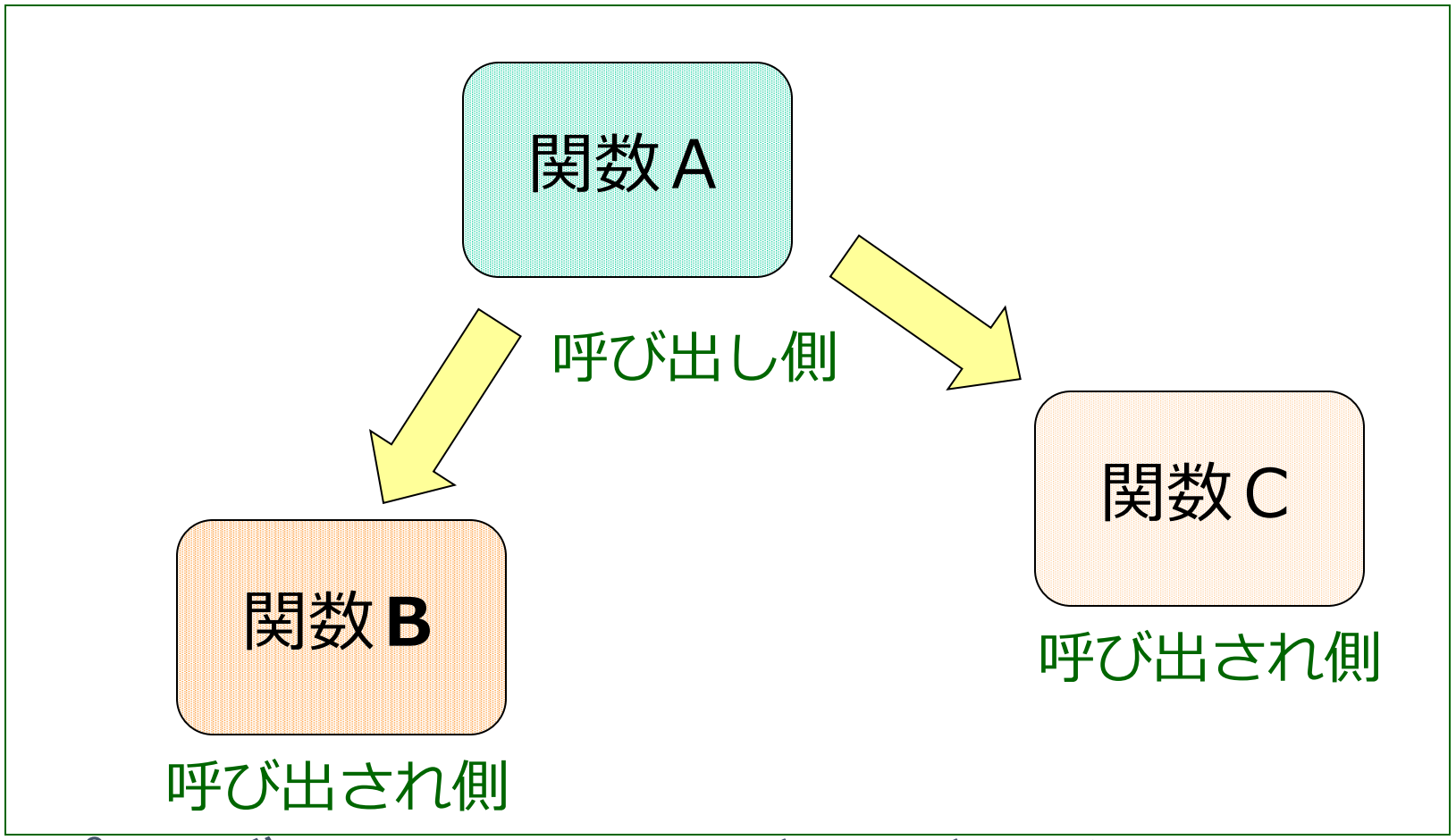
例題５．カレンダー

　　関数の集まりとしてのプログラム

仕事の分割



関数とは



プログラムは，しばしば，複数の関数に「分割」される

目標

- １つの main 関数と，その他の複数の関数から構成されたプログラムを読んで，理解できる能力を習得する

- 複数の関数の間での「情報の受け渡し」について理解する

例題１．棒グラフ

- 整数から，その長さだけの棒を表示する関数を作る

  例） 5 → *****

- 上記で作った関数を使って，「整数を読み込んで，読み込んだ長さの棒を表示するプログラム」を作る

棒グラフ

```
#include <stdio.h>
#pragma warning(disable:4996)
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
```

bar関数

```
int main()
{
    int len;
    printf( "len=" );
    scanf( "%d", &len );
    bar( len );
    return 0;
}
```

main関数

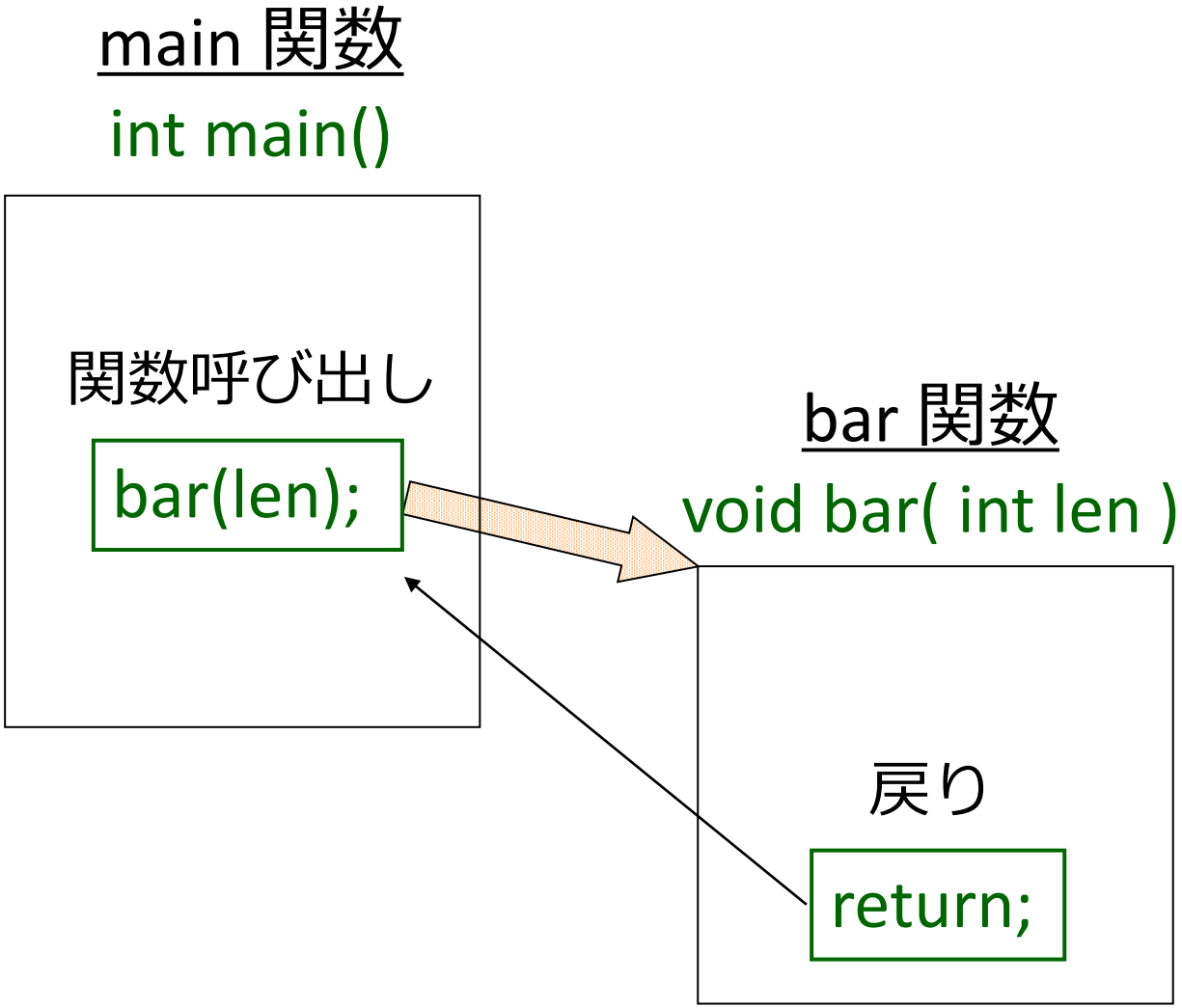
棒グラフ

実行結果の例

```
len=5

*****
```

関数呼び出しの流れ


main 関数
int main()
関数呼び出し
bar(len);
bar 関数
void bar(int len)
戻り
return;

プログラム実行順

```
#include <stdio.h>
#pragma warning(disable:4996)
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
④       printf("*");
    }
⑤   printf("¥n");
⑥   return;  戻り
}

int main()
{
    int len;
①   printf( "len=" );
②   scanf( "%d", &len );
③   bar( len );  関数呼び出し
⑦   return 0;
}
```

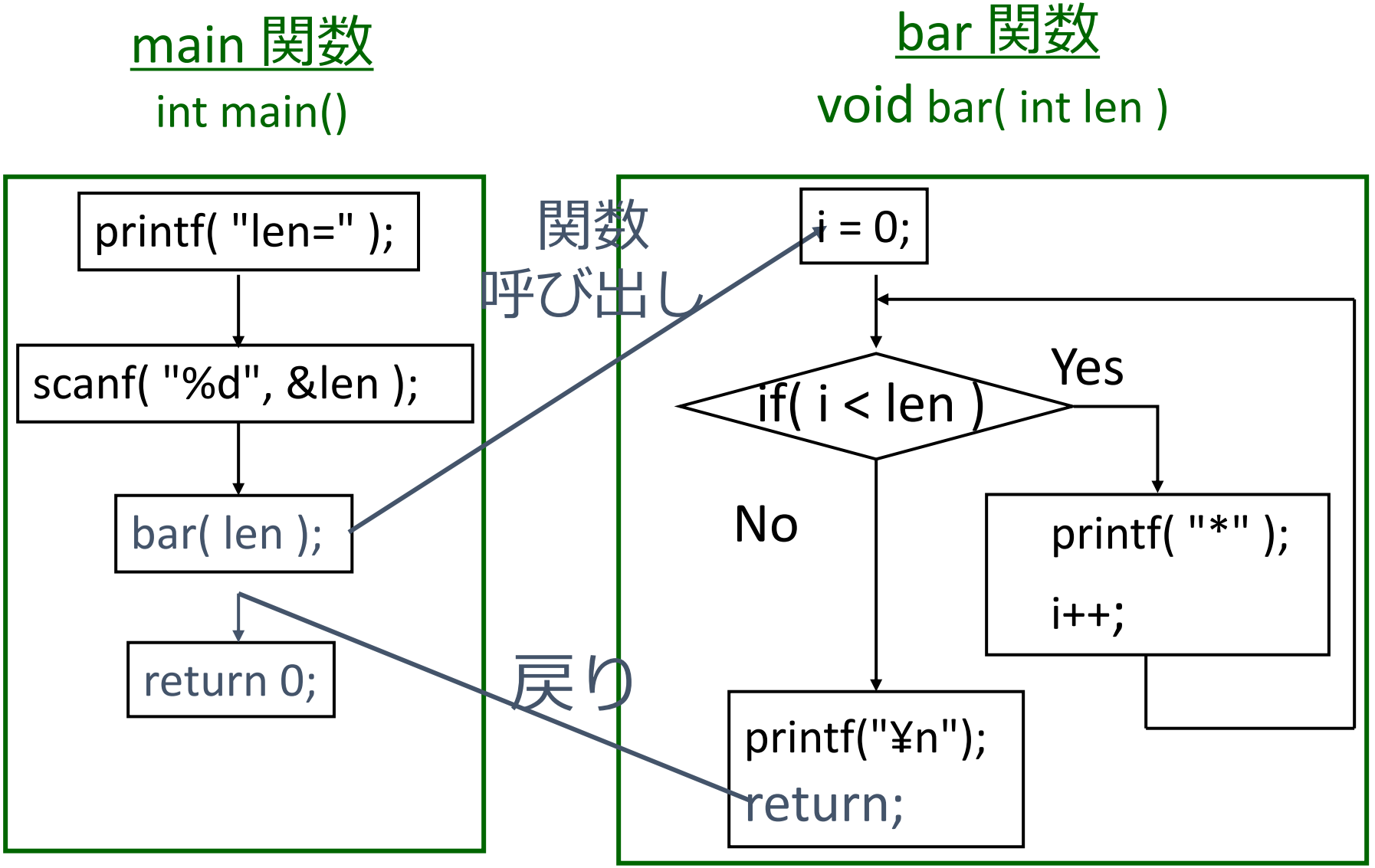
bar関数

main 関数の先頭行
がプログラムの始まり

main関数

main 関数内の return
がプログラムの終わり

プログラム実行順


main 関数
int main()
bar 関数
void bar(int len)
printf("len=");
scanf("%d", &len);
bar(len);
return 0;
関数
呼び出し
戻り
i = 0;
if(i < len)
Yes
No
printf("*");
i++;
printf("¥n");
return;

プログラム実行順

- 普通，プログラム中の文は逐次的に実行される
- 関数呼び出しでは，関数の先頭に「ジャンプ」する.

  関数呼び出しの例） bar(len);

- 関数の中で return 文に出会うと，関数呼び出しの場所に戻る.

main 関数

- プログラムの実行を開始すると，自動的にmain関数が呼ばれる
  - プログラムの開始は，main 関数の最初の行から
  - main 関数内の return 文は，「プログラムの終わり」を示す
- プログラムには，必ず main 関数が書かれていなければならない

関数定義の例

```
#include <stdio.h>
#pragma warning(disable:4996)
void bar( int len )
{
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}

int main()
{
    printf( "len=" );
    scanf( "%d", &len );
    bar( len );
    return 0;
}
```

型 名前 仮引数

型 名前 仮引数（は空）

頭部

本体

bar関数

頭部

本体

main関数

関数定義

- 関数定義とは，関数の実体をプログラムとして書くこと
- 関数には，名前，型，仮引数がある

例）
```
void bar( int len )
int  main()
```

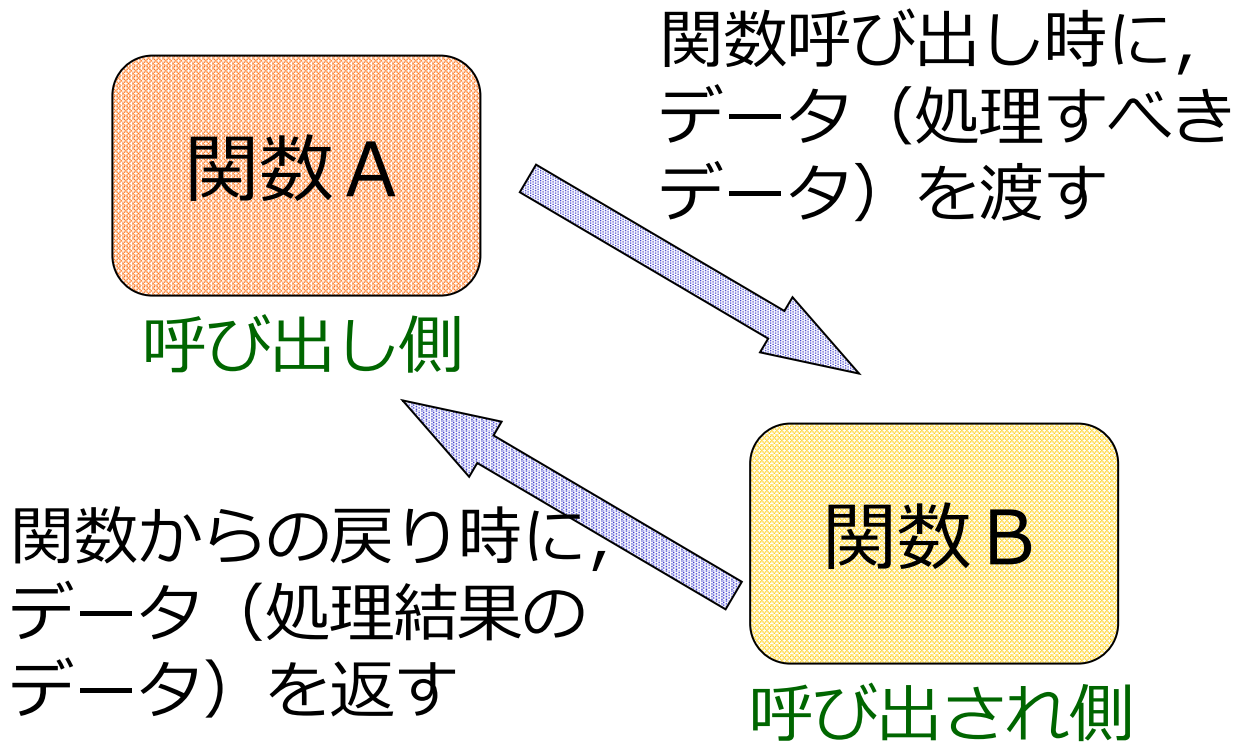
型　名前　仮引数

関数側から呼び出し側に返されるデータに関係する

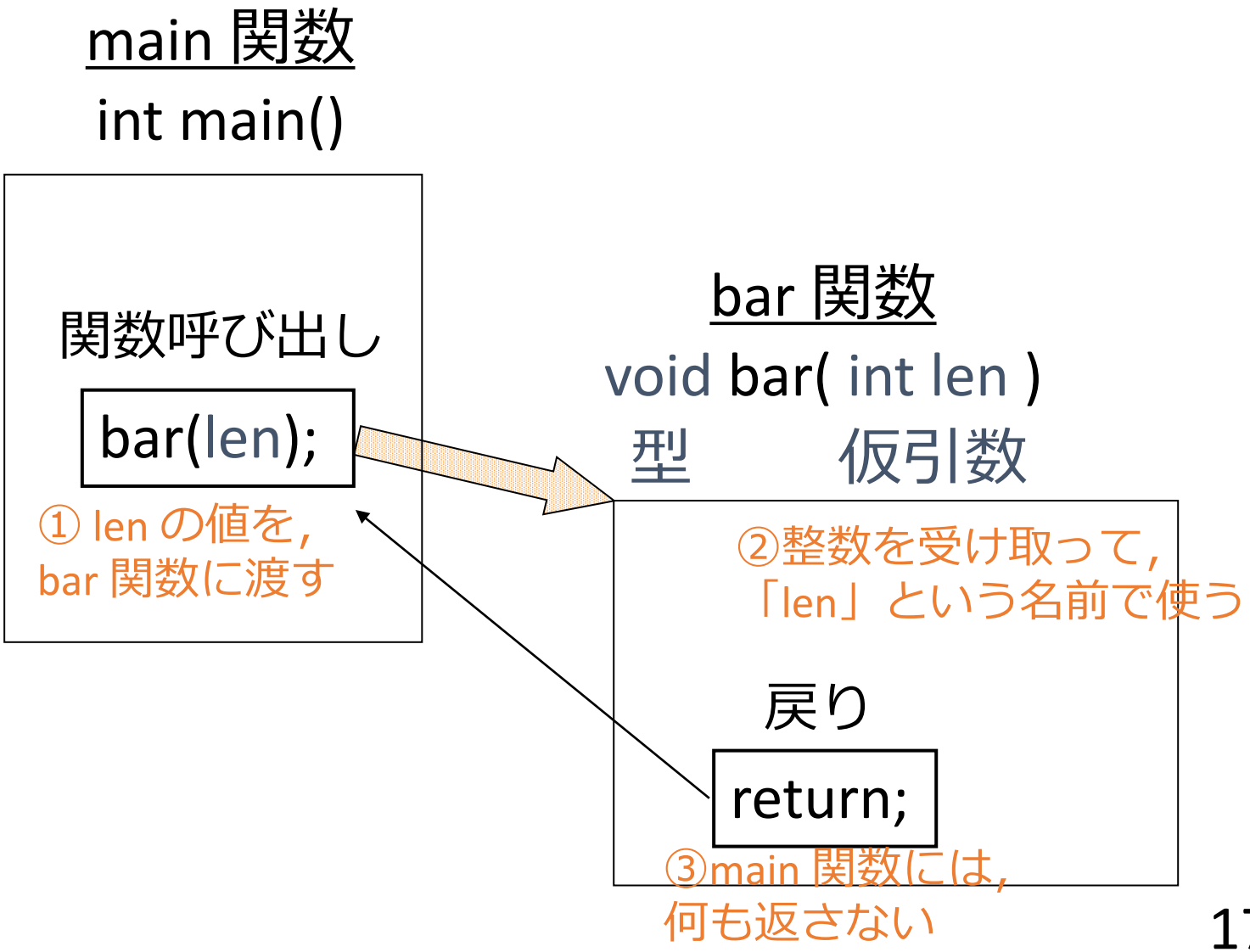
呼び出し側から関数側に渡されるデータに関係する

- 仮引数のそれぞれにも，型と名前がある

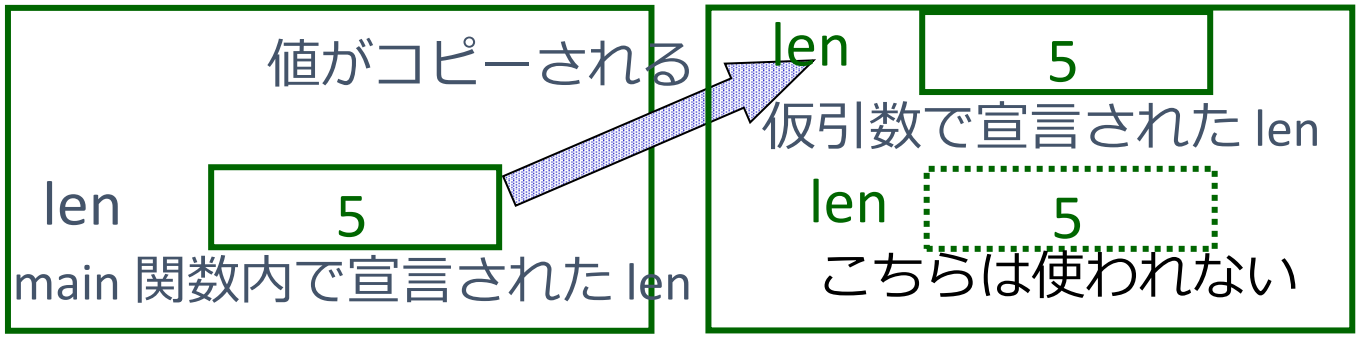
関数でのデータの流れ


関数A
呼び出し側
関数呼び出し時に，
データ（処理すべき
データ）を渡す
関数からの戻り時に，
データ（処理結果の
データ）を返す
関数B
呼び出され側

データの流れ


main 関数
int main()
関数呼び出し
bar(len);
① len の値を，
bar 関数に渡す
bar 関数
void bar(int len)
型
仮引数
②整数を受け取って，
「len」という名前で使う
戻り
return;
③main 関数には，
何も返さない

データの流れ



main 関数内で宣言された len と，仮引数で宣言された len は別のもの

main 関数

```
int main()
```

関数呼び出し

```
bar(len);
```

① len の値を，bar 関数に渡す

sum 関数

```
void bar( int len )
```

型　　仮引数

②整数を受け取って，「len」という名前で使う

戻り

```
return;
```

③main 関数には，何も返さない

引数と仮引数

- 関数呼び出し側
  - カッコの中に書いた変数等の値（引数という）が，関数に渡される

（例）　bar(len);

len の値を，bar 関数に渡す（引数）

- 呼び出された関数側
  - 呼び出された関数は，値を受け取って，名前を付けて使用する（仮引数という）

（例）　int bar(int len)

整数を受け取って，「len」という名前で使う（仮引数）

引数と仮引数

- 引数
  - 関数などに実際に渡される「データ」のこと
  - 関数呼出しの（，）内に並べる
  - 定数や変数や式を書く

- 仮引数（パラメータともいう）
  - 関数定義の（，）中で宣言された変数のこと.
    （例） **void bar(int len)**では，変数 len が宣言されている.
  - 関数は，呼び出された時点において，値を受け取り，当該関数の仮引数である変数にセットする.

- 引数で与えられた値が，仮引数の変数に代入されて情報の受け渡しが行われる.

例題２．月の日数

- 年と月から，日数を求める関数を作る
  - うるう年の２月ならば２９
  - 求めた日数は，関数の返り値として，呼び出し側に返すこと.

    例） 2001 年 11 月 → 30

- 上記で作った関数を使って，「年と月を読み込んで，日数を求めるプログラム」を作る

```
#include <stdio.h>
#pragma warning(disable:4996)
int num_of_day( int y, int m)
{
    int num_days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if ( (m == 2) && (((y % 400) == 0) || (((y % 100) != 0) && ((y % 4) ==
   0)))){
        return 29;
    }
    else {
        return num_days[m-1];
    }
}
int main()
{
    int y;
    int m;
    int n;
    printf( "y=" );
    scanf( "%d", &y );
    printf( "m=" );
    scanf( "%d", &m );
    n = num_of_day(y, m);
    printf( "number of days(%d) = %d¥n", m, n );
    return 0;
}
```

月の日数

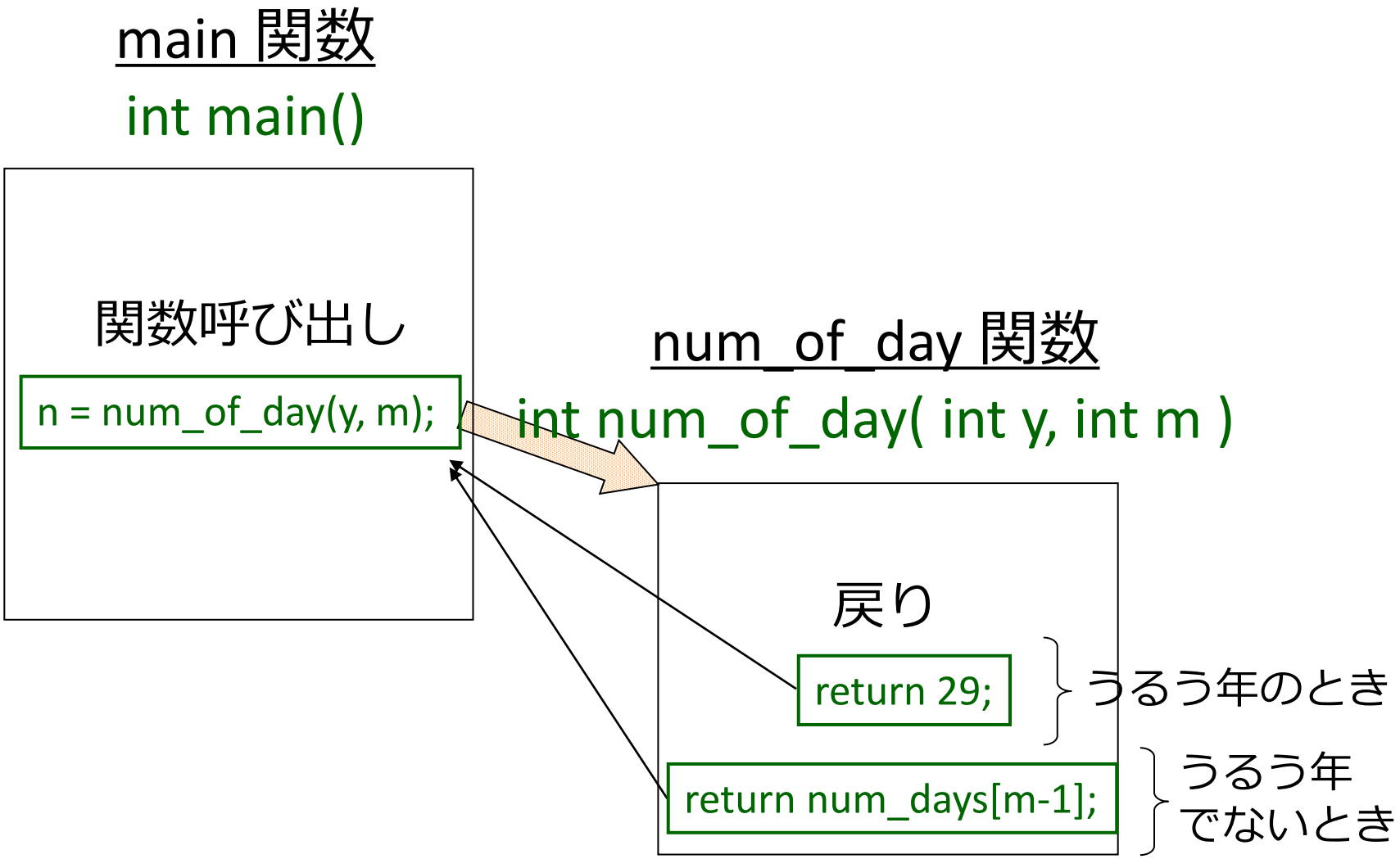
実行結果の例

```
y=2001

m=11

number of days(11) = 30
```

関数呼び出しの流れ


main 関数
int main()
関数呼び出し
n = num_of_day(y, m);
num_of_day 関数
int num_of_day(int y, int m)
戻り
return 29;
うるう年のとき
return num_days[m-1];
うるう年
でないとき

プログラム実行順


```
#include <stdio.h>
#pragma warning(disable:4996)
int num_of_day( int y, int m)
{
    int num_days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
⑤  if ( ( m == 2 ) && (((y % 400) == 0) ||  (((y % 100) != 0) && ((y %4) == 0)))){
⑥      return 29;
    }
    else {
⑥      return num_days[m-1];
    }
}
int main()
{
    int y;
    int m;
    int n;
①  printf( "y=" );
②  scanf( "%d", &y );
③  printf( "m=" );
④  scanf( "%d", &m );
⑦  n = num_of_day(y, m);
⑧  printf( "number of days(%d) = %d¥n", m, n );
⑨  return 0;
}
```

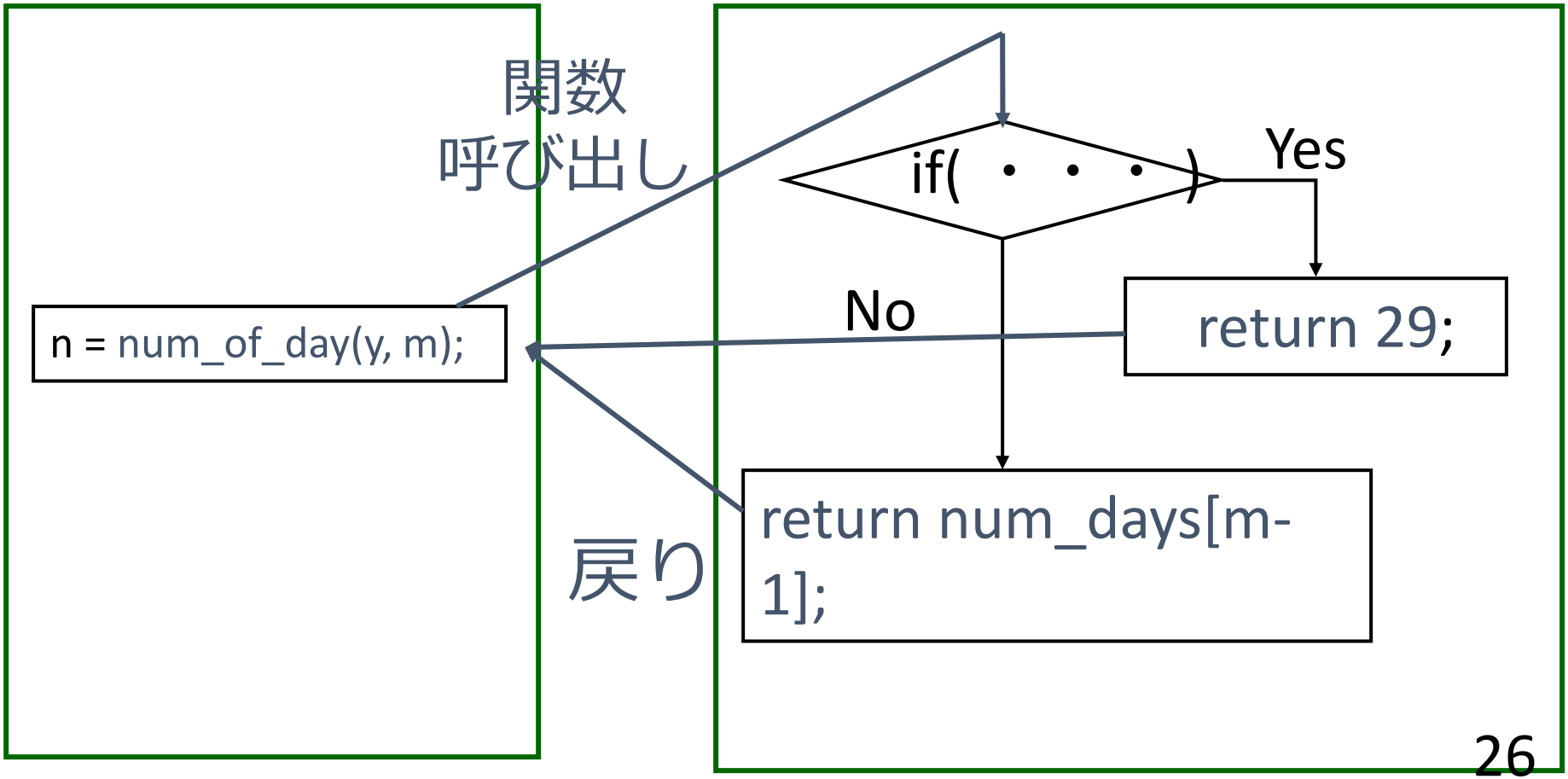
プログラム実行順

main 関数

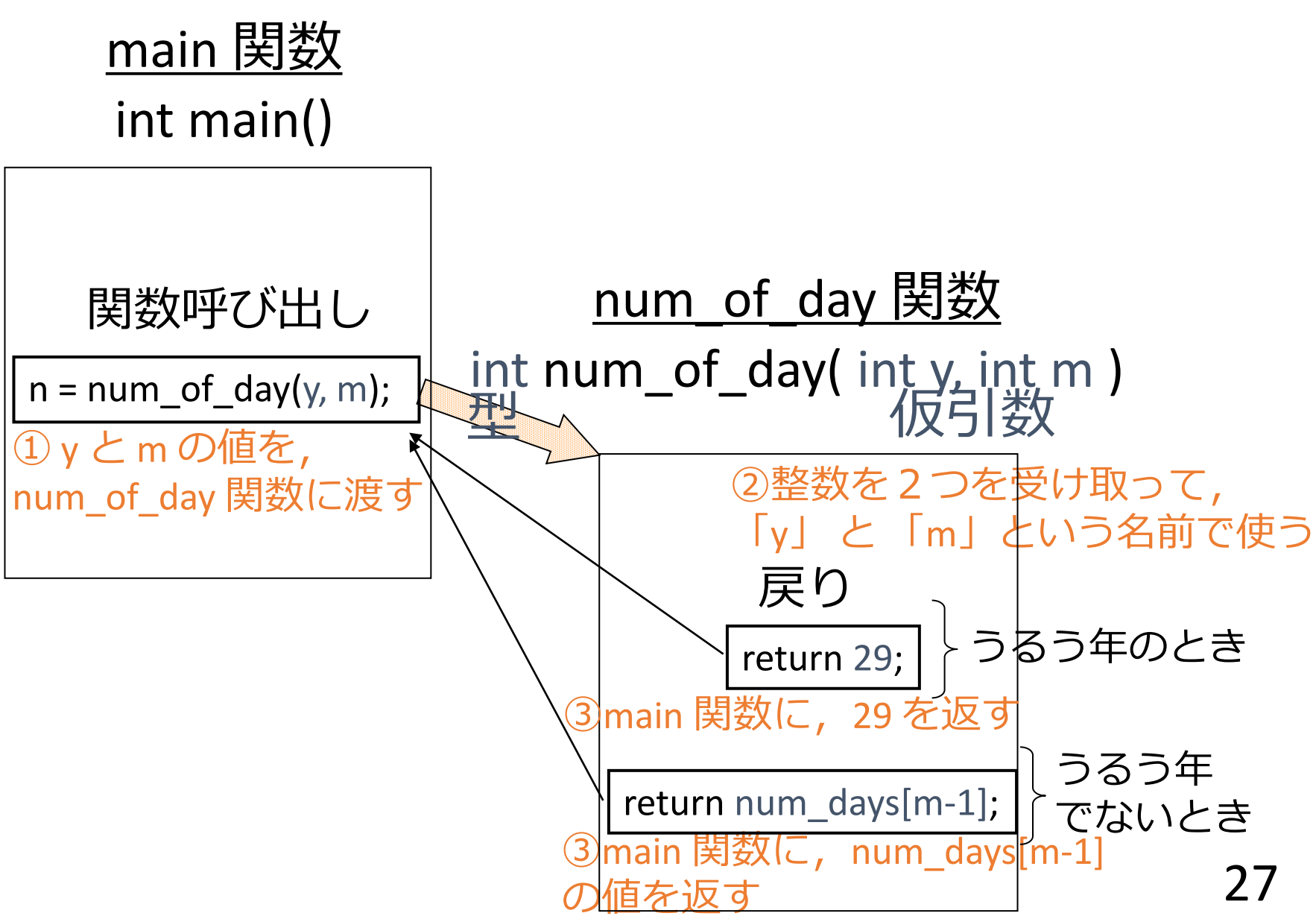
int main()

num_of_day 関数

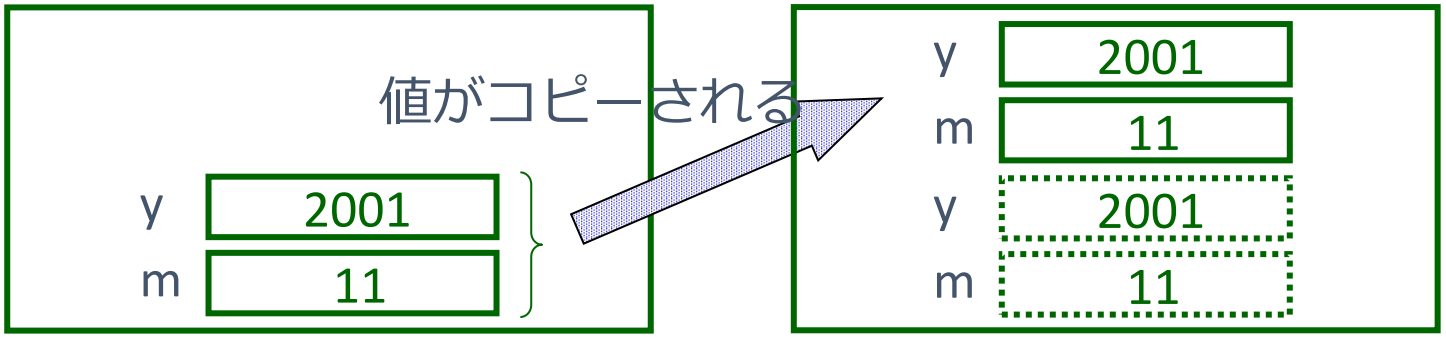
int num_of_day(int y, int m)



データの流れ


main 関数
int main()
関数呼び出し
n = num_of_day(y, m);
① y と m の値を，
num_of_day 関数に渡す
num_of_day 関数
int num_of_day(int y, int m)
型
仮引数
②整数を 2 つを受け取って，
「y」と「m」という名前で使う
戻り
return 29;
うるう年のとき
③main 関数に，29 を返す
return num_days[m-1];
うるう年
でないとき
③main 関数に，num_days[m-1]
の値を返す

データの流れ



main 関数

int main()

関数呼び出し

n = num_of_day(y, m);

① y と m の値を，
num_of_day 関数に渡す

num_of_day 関数

int num_of_day(int y, int m)

型　仮引数

②整数を 2 つを受け取って，
「y」 と 「m」という名前で使う

戻り

return 29;

うるう年のとき

③main 関数に，29 を返す

return num_days[m-1];

うるう年
でないとき

③main 関数に，num_days[m-1]
の値を返す

関数の中の return 文

- 関数の呼び出しの場所に戻ることを示す

- return 文で書いた式の値が，呼び出し側に返される

（例）return;　　← 何も返さない

return 29;　　← 29 を返す

return num_days[m-1]; ← num_days[m-1] を返す

void の意味

- 関数には，型がある
  - int
  - double
  - void など
  - 但し，void は，「関数が return 文で，何も返さないこと（値を返さない関数）」 を示す

    例） void print_calendar(int num_days, int youbi)

- void で関数定義したら，return の次に式を書かない（ 「return;」 ）.

関数から返された値の使い方

```
n = num_of_day(y, m);
```

関数呼び出し

num_of_day 関数から返された値が n に入る

- 関数は，一種の「式」であって，「値」を持つ
  - この「値」が，関数から返された値

例題３．１か月分のカレンダー

- 日数と曜日から，１か月分のカレンダーを表示する関数を作る
  - 日数は，２８，２９，３０，３１ のいずれか
  - 曜日は，０，１，…，６ のいずれか
  - 表示される日付は，２桁の幅とし，それぞれの間に１つの空白を置くこと.

```
#include <stdio.h>
#pragma warning(disable:4996)

void print_calendar( int num_days, int youbi )
{
    int i;
    int d;
    int x;
    if ( ( num_days < 28 ) || ( num_days > 31 ) || ( youbi < 0 ) || ( youbi > 6 ) )
    {
        return;
    }

    for ( i = 0; i < ( youbi * 3 ); i++ ) {
        printf( " " );
    }
    d = 1;
    x = youbi;
    do {
        printf( "%2d ", d );
        d++;
        if ( x == 6 ) {
            printf( "¥n" );
            x = 0;
        }
        else {
            x++;
        }
    } while ( d <= num_days );
    return;
}
```

渡された値がおかしいときは，何もせずに終わる

曜日の分だけ空白を表示

日付を書く

土曜日に来たら改行する

月末まで書いたら終える

前のページからの続き

```
int main()
{
    int num_days;
    int youbi;
    printf( "num_days=" );
    scanf( "%d", &num_days );
    printf( "youbi=" );
    scanf( "%d", &youbi );
    print_calendar( num_days, youbi );
    return 0;
}
```

１か月分のカレンダー

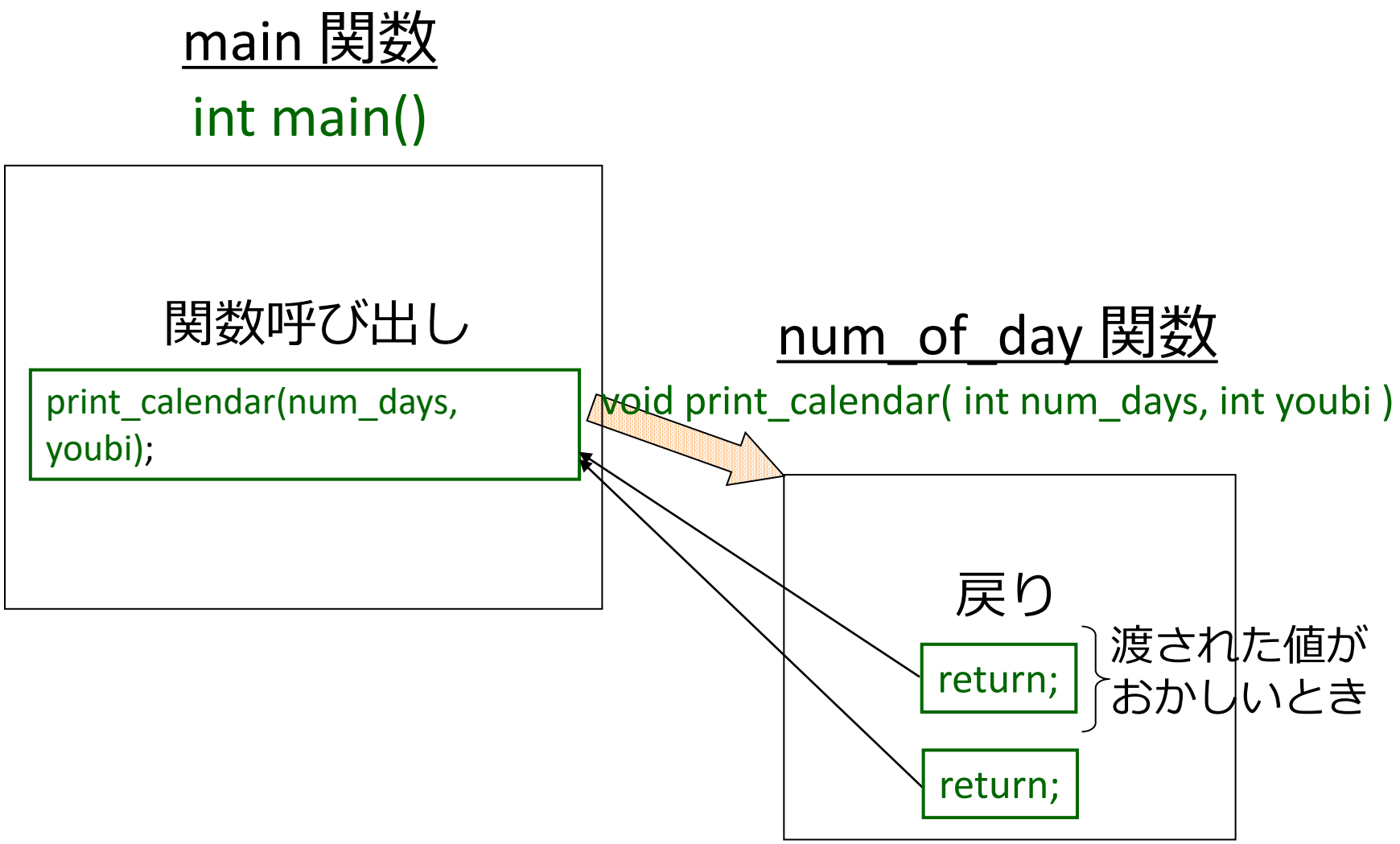
実行結果の例

```
num_days=30

youbi=4

                1  2  3
4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

関数呼び出しの流れ


main 関数
int main()
関数呼び出し
print_calendar(num_days, youbi);
num_of_day 関数
void print_calendar(int num_days, int youbi)
戻り
return;
return;
渡された値がおかしいとき

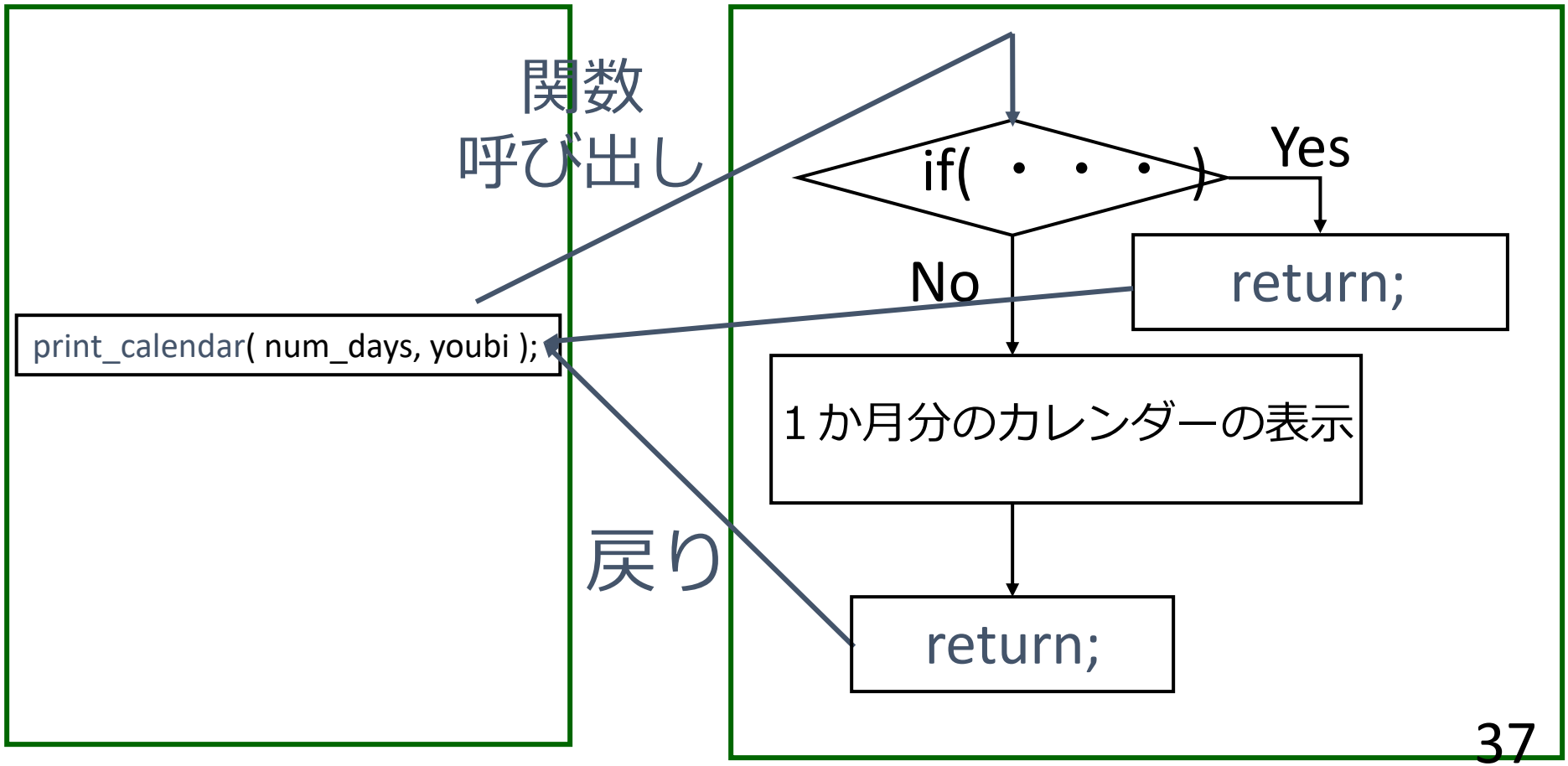
プログラム実行順

main 関数

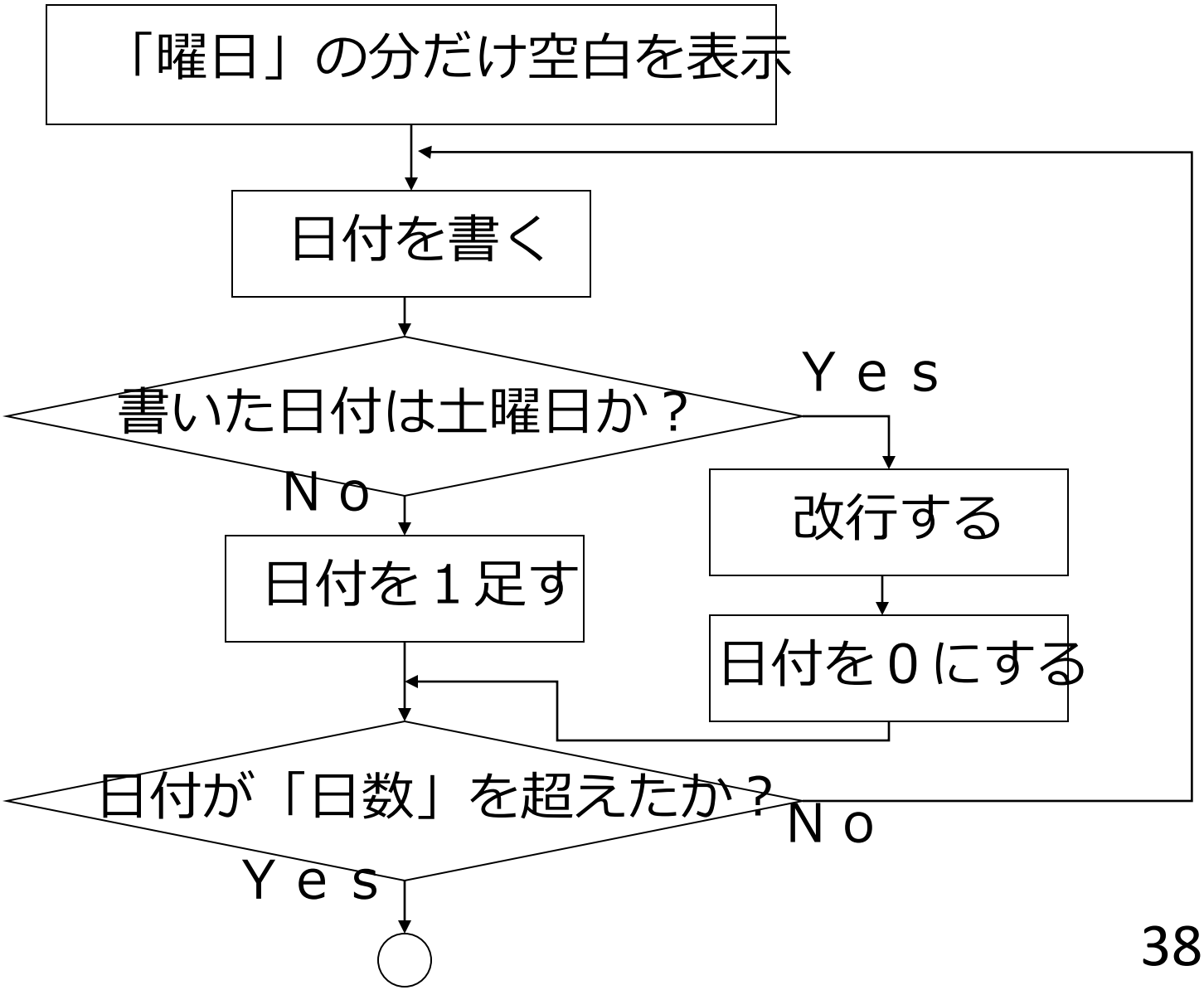
int main()

print_calendar 関数

void print_calendar(int num_days, int youbi)



1か月分のカレンダーの表示



例題４．月初めの曜日

- ツエラーの公式を使い，年と月から，月初めの曜日を求める関数を作る
  - 曜日は，０，１，…，６ のいずれか

０：日曜日
１：月曜日
２：火曜日
３：水曜日
４：木曜日
５：金曜日
６：土曜日

月初めの曜日

```
#include <stdio.h>
#pragma warning(disable:4996)

int zeller( int y, int m, int d )
{
    if ( ( m == 1 ) || ( m == 2 ) ) {
        y = y - 1;
        m = m + 12;
    }
    return (y + (y/4) - (y/100) + (y/400) + ((13 * m + 8 ) / 5) + d) % 7;
}
int first_day( int y, int m )
{
    return zeller( y, m, 1 );
}
int main()
{
    int y;
    int m;
    int f;
    printf( "y=" );
    scanf( "%d", &y );
    printf( "m=" );
    scanf( "%d", &m );
    f = first_day(y, m);
    printf( "first days(%d, %d) = %d¥n", y, m, f);
    return 0;
}
```

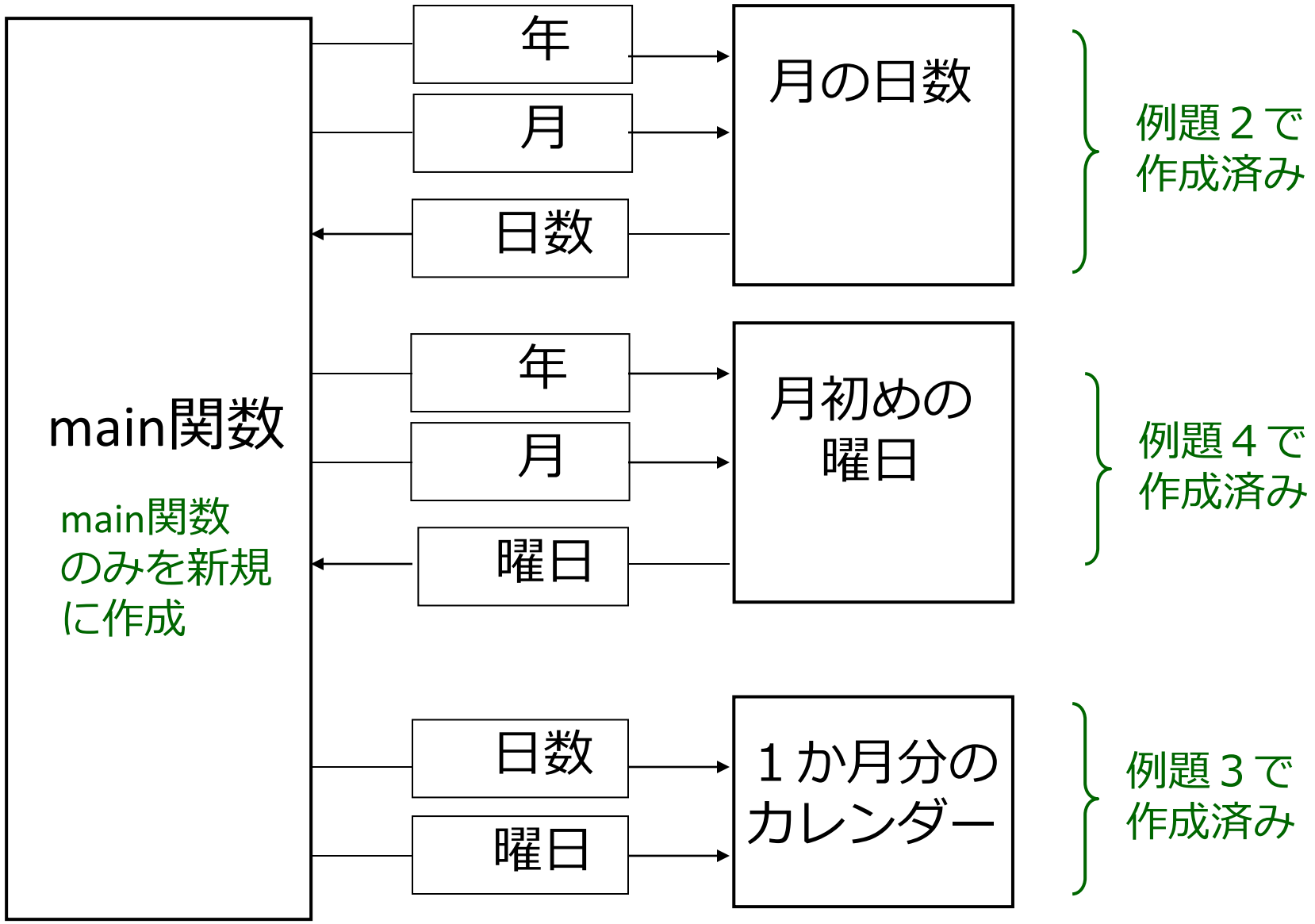
例題５．カレンダー

- 年と月から，カレンダーを表示するプログラムを作る

「カレンダー」のプログラムの関数分割

- 役割ごとに，関数を作る．
  1. 本体
  2. 月の日数
  3. 月初めの曜日
  4. １ヶ月分のカレンダー

カレンダー


main関数
main関数
のみを新規
に作成
年
月
日数
月の日数
例題２で
作成済み
年
月
曜日
月初めの
曜日
例題４で
作成済み
日数
曜日
１か月分の
カレンダー
例題３で
作成済み

コメントでおおまかな処理を書く

```
int main()
{
   /* 年と月を読み込む */
   /* 年と月から「月の日数」を求める */
   /*年と月から「月初めの曜日」を求める */
   /* 「月の日数」と「月初めの曜日」からカレンダーを表示する */
   return 0;
}
```

コメント

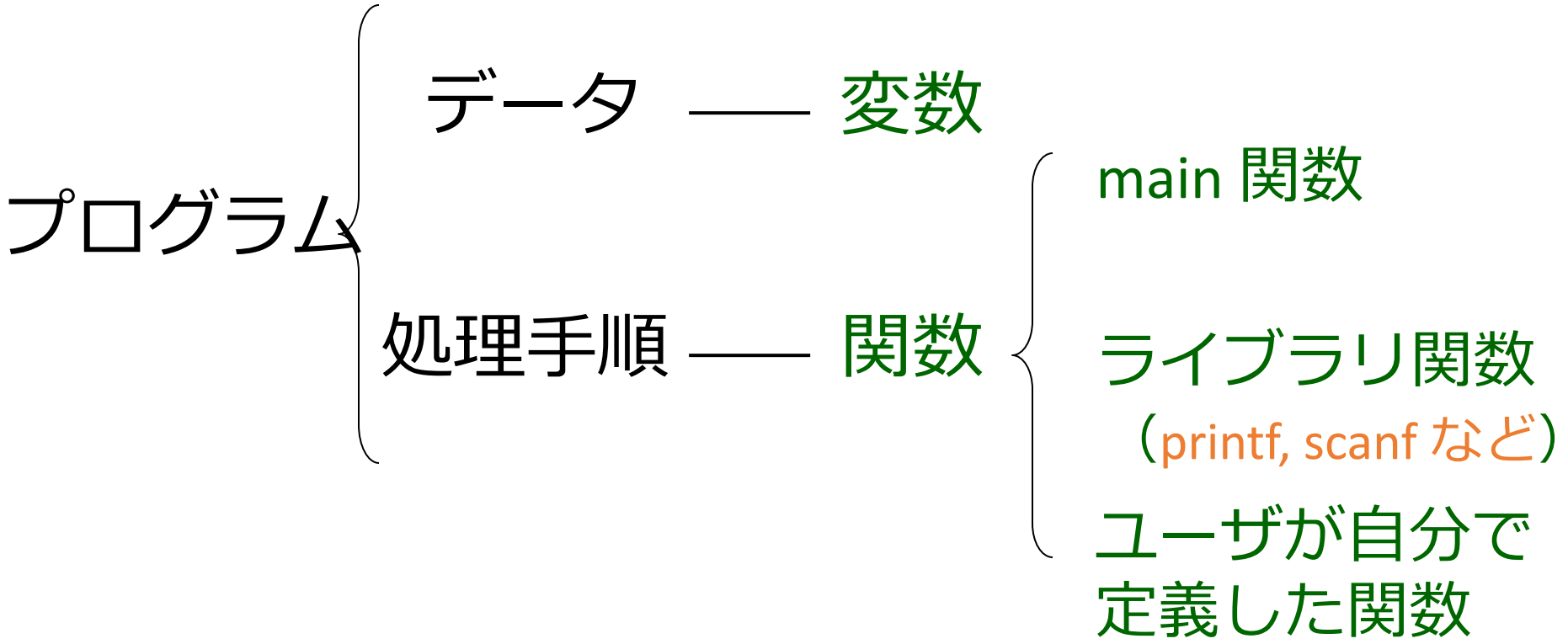
- コメントは，プログラムの中に書く注釈のこと.
- プログラムの説明や使用上の注意などを書く.
- コメントは，プログラム実行では無視される.
- コメントの始まりは／＊で，終わりは＊／である．あるいは／／を使って，１行分のコメントを書くこともある．コメントは，入れ子にすることができない.
- 次のコメントは入れ子になっていて，間違いである.

```
/* n = n + 1; /* n は「空き領域」の先頭を指す */ */
```

カレンダー

```
#pragma warning(disable:4996)
int main()
{
    int y;
    int m;
    int nissu;
    int youbi1;
    /* 年と月を読み込む */
    printf( "y=" );
    scanf( "%d", &y );
    printf( "m=" );
    scanf( "%d", &m );
    printf( "日 月 火 水 木 金 土¥n" );
    /* 年と月から「月の日数」を求める */
    nissu = num_of_day(y, m);
    /*年と月から「月初めの曜日」を求める */
    youbi1 = first_day(y, m);
    /* 「月の日数」と「月初めの曜日」からカレンダーを表示する */
    print_calendar( nissu, youbi1 );
    return 0;
}
```

－関数と変数－


プログラム
データ ―― 変数
処理手順 ―― 関数
main 関数
ライブラリ関数
（printf, scanf など）
ユーザが自分で
定義した関数

関数の種類

- **main関数**
  - プログラムは，main 関数を必ず含む
- ライブラリ関数
  - システムに既に組み込まれた関数
- ユーザが定義した関数
  - プログラマが独自の関数を定義可能