プログラミング基礎

http://bit.ly/prog2d

引数とポインタ

後期 第3週

2017/10/10

今回は

「2つの変数の値を交換する」処理を通して、関数の引数にポインタを利用する方法について説明します

① 交換する処理をmain()で作る

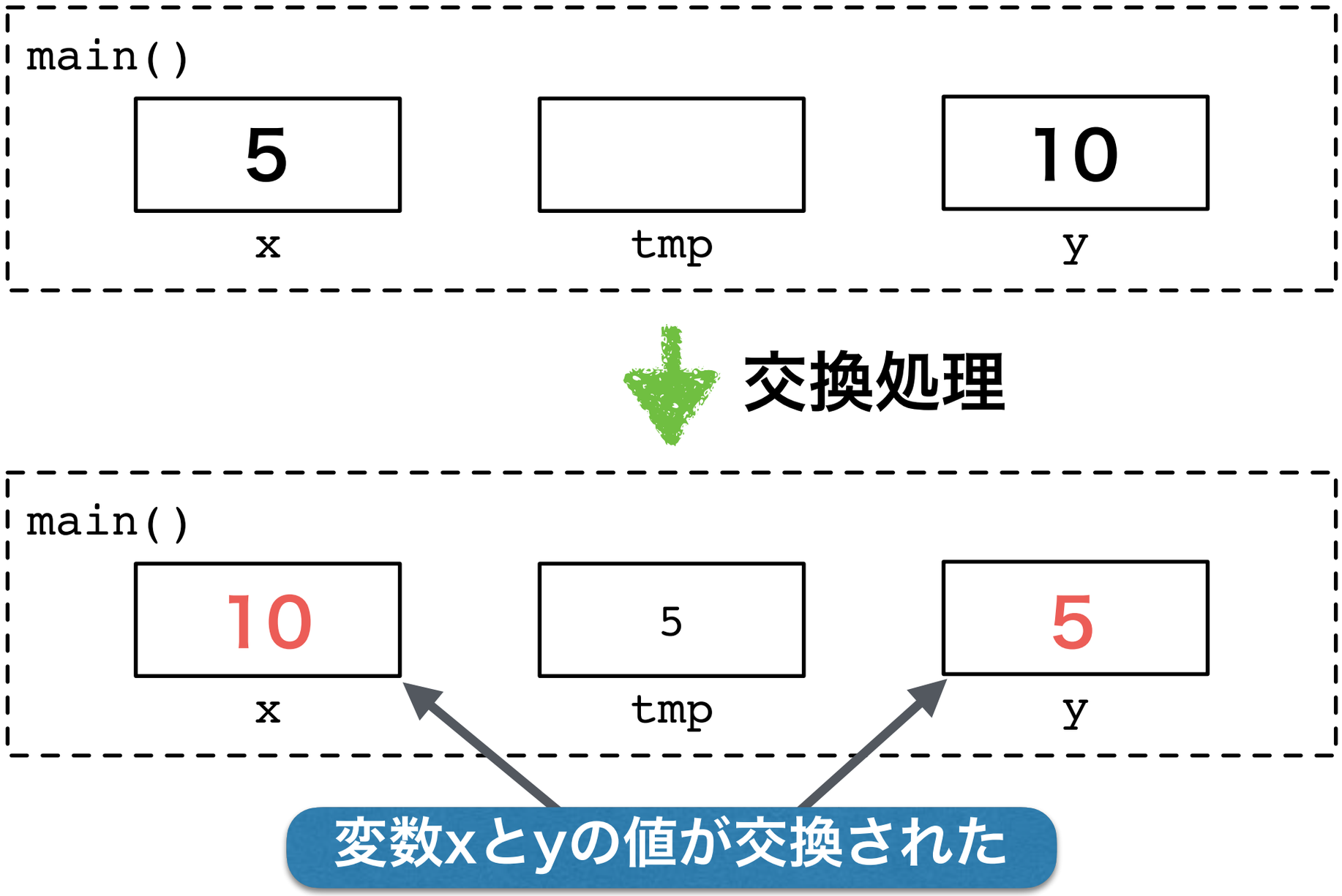
【例1】

```
int main(void)
{
    int x, y, tmp;
    x = 5;
    y = 10;
    printf("x: %d, y: %d\n", x, y);
    tmp = x;
    x = y;
    y = tmp;
    printf("x: %d, y: %d\n", x, y);

    return 0;
}
```

変数xとyの値を交換する

① 交換する処理をmain()で作る

【例1のイメージ】



② 交換する処理の関数を作る

【例2】

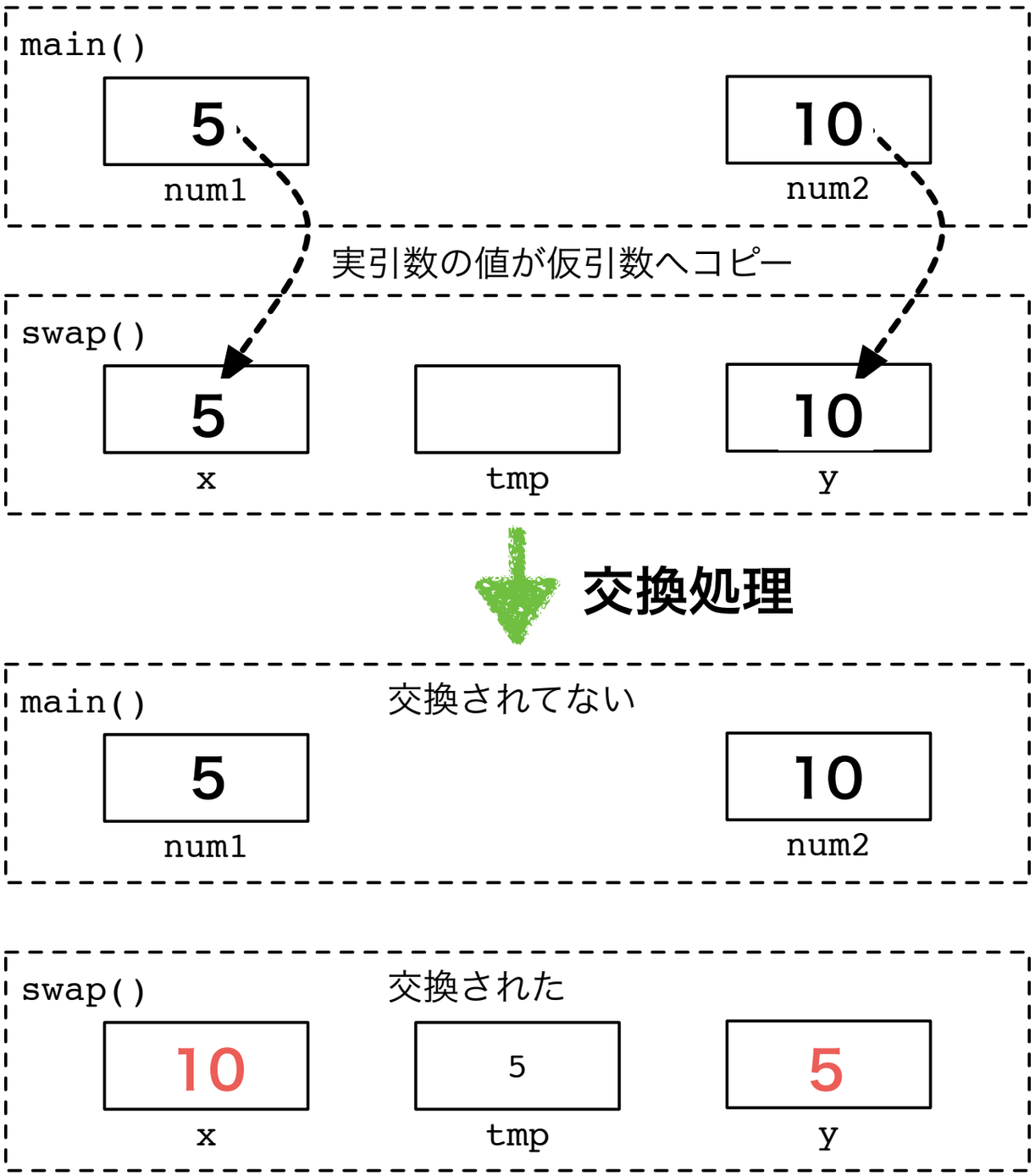
```
#include <stdio.h>
void swap(int x, int y);
int main(void)
{
    int num1, num2;
    num1 = 5;
    num2 = 10;
    printf("num1: %d, num2: %d\n", num1, num2);
    swap(num1, num2);
    printf("num1: %d, num2: %d\n", num1, num2);
    return 0;
}

void swap(int x, int y)
{
    int tmp;
    tmp = x;
    x = y;
    y = tmp;
}
```

num1とnum2を実引数にしてswapを呼び出す

仮引数xとyの値を交換する

② 交換する処理の関数を作る

【例2のイメージ】



仮引数の値が交換されても、**実引数の値は交換されない** (p.289)

③ 仮引数をポインタにする

【例3】

```
#include <stdio.h>
void swap(int *pX, int *pY);
int main(void)
{
    int num1, num2;
    num1 = 5;
    num2 = 10;
    printf("num1: %d, num2: %d\n", num1, num2);
    swap(&num1, &num2);
    printf("num1: %d, num2: %d\n", num1, num2);
    return 0;
}

void swap(int *pX, int *pY)
{
    int tmp;
    tmp = *pX;
    *pX = *pY;
    *pY = tmp;
}
```

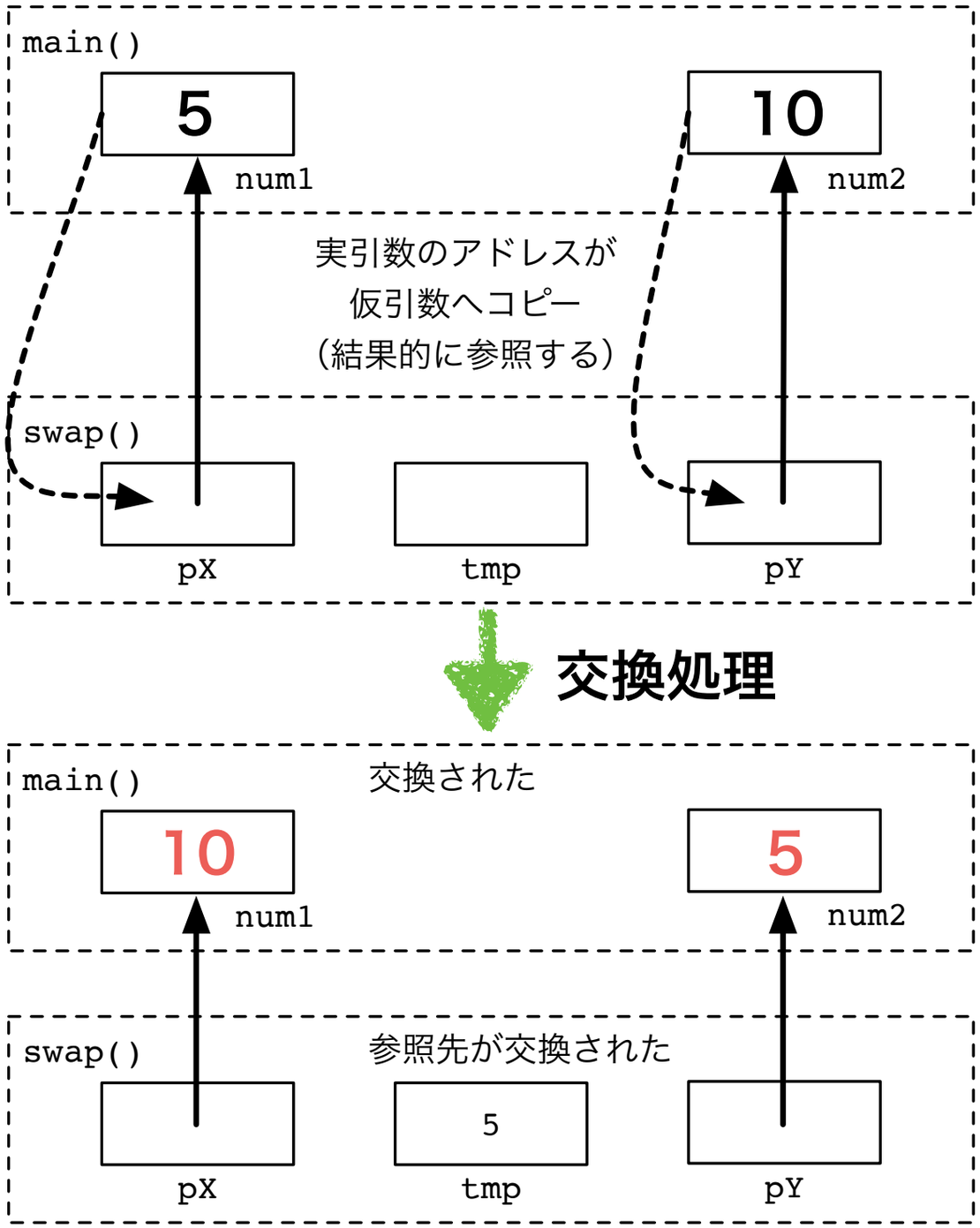
num1とnum2のアドレスを引数にしてswapを呼び出す

仮引数をポインタにする

ポインタpXとpYの参照先を交換する

③ 仮引数をポインタにする

【例3のイメージ】



仮引数の参照先の値が交換されると、mainの変数の値も交換される
(p.292)

話をまとめると・・・

- 例2のように、実引数の値が仮引数へコピーされる渡し方を値渡しといいます。
- 例3のように、実引数のアドレスが仮引数のポインタへコピーされる渡し方を参照渡しといいます。
- 戻り値の仕組み（関数内でreturnを使うこと）を利用すると、関数内で変更した値を1つしか返すことができませんが、参照渡しを利用することで、関数内で複数個の変数の値を変更して、関数を呼び出した側に戻ることができるようになります。（p.294）
- 前期に作成した関数の場合、関数の中でも外でも使用したい変数や配列は、グローバル変数で宣言していたので、その関数はその変数や配列に固定された処理になってしまいました。参照渡しを利用することで、グローバル変数で宣言する必要もなくなり、関数が処理する対象となる変数や配列が固定されず、より汎用的な関数を作ることができるようになります。

実引数を変更できないようにする方法

仮引数にポインタを使っていても、実引数の値を変更したくない場合は、仮引数に「const」を付加することで明示的に変更を防ぐことができます。

```
void func(const int *pX){}
  /* ポインタpxの参照先の値を変更できなくなる */
```

【練習3-1】

例2と例3にあるプログラムを実行してみましょう。

その際、仮引数の交換処理の変化もわかるように、
出力処理を追加してください。（次頁の実行例参照）

【練習3-1】

[例2の実行結果]

```
(交換前) num1: 5, num2: 10
(交換前) x: 5, y: 10
(交換後) x: 10, y: 5
(交換後) num1: 5, num2: 10
```

[例3の実行結果]

```
(交換前) num1: 5, num2: 10
(交換前) *pX: 5, *pY: 10
(交換後) *pX: 10, *pY: 5
(交換後) num1: 10, num2: 5
```

【練習3-2】

例3のプログラムに対して、仮引数のポインタpYの参照先の値を変更できないように関数swap()を変更してください。

変更すると、コンパイル時に以下のようなメッセージが出力されます。

[コンパイル時に出力されるエラーメッセージの例]

```
error: read-only variable is not assignable
    *pY = tmp;
    ~~~ ^
```

【課題3-1】

仮引数で与えられた3つのポインタが参照する整数値の合計を求め、1つ目の仮引数にその合計が入る関数add_num()を作成してください。

[この関数のプロトタイプ宣言]

```
void add_num(int *num1, int *num2, int *num3);

  /* *num1に、*num1～*num3の合計を代入する */
  /* *num2, *num3は0にリセットされる */
```

【課題3-1】

[mainでの処理]

```
 int n1, n2, n3;
 n1 = 5; n2 = 7; n3 = -3;
 add_num(&n1, &n2, &n3);
 printf("n1: %d, n2: %d, n3: %d\n", n1, n2, n3);
```

[実行結果]

```
 n1: 9, n2: 0, n3: 0
```

【課題3-2】

仮引数で与えられた3つのポインタが参照する整数値の絶対値の合計を求め、1つ目の仮引数にその合計が入る関数add_absolute()を作成してください。

[この関数のプロトタイプ宣言]

```
void add_absolute(int *num1, int *num2, int *num3);

  /*  *num1～*num3の絶対値を求めて、合計を*num1に代入する  */
  /*  *num2, *num3は0にリセットされる  */
```

【課題3-2】

[mainでの処理]

```
 n1 = 6; n2 = -8; n3 = -9;
 add_absolute(&n1, &n2, &n3);
 printf("n1: %d, n2: %d, n3: %d\n", n1, n2, n3);
```

[実行結果]

```
 n1: 23, n2: 0, n3: 0
```

【課題3-3】

仮引数dateで与えられた「年月日」を表す8桁の整数に対して、年, 月, 日に分けた値を求める関数divide_date()を作成してください。

[この関数のプロトタイプ宣言]

```
void divide_date(int date, int *y, int *m, int *d);

  /* 年を表す値は*y、月は*m、日は*dに代入される */
  /* dateを100で割った余りで*dが求まる */
  /* dateを100で割って下2桁を切り捨てる */
  /* 同様の処理を繰り返すと*m, *yも求まる */
```

【課題3-3】

[mainでの処理]

```
 int date1, y1, m1, d1;
 date1 = 20121015;
 divide_date(date1, &y1, &m1, &d1);
 printf("%d / %d / %d\n", y1, m1, d1);
```

[実行結果]

```
 2012 / 10 / 15
```

【課題3-4】

仮引数vで与えられた浮動小数点数に対して、「整数値に切り上げた値」、「整数値に切り捨てた値」、「四捨五入した値」を求める関数round_value()を作成してください。

【課題3-4】

[この関数のプロトタイプ宣言]

```
void round_value(float v, int *roundup, int *rounddown,
                 int *round);

  /* 切り上げた値は*roundup、切り捨てた値は*rounddown、
        四捨五入した値は*roundに代入される */
  /* ● vをint型にキャスト（型変換）すると*rounddownが求まる */
  /* ● vが*rounddownよりも大きい場合、
        *roundupは*rounddownよりも1大きい値
        (そうでない場合は、*rounddownと同じ値) */
  /* ● vと*rounddownの差が0.5以上の場合、
        *roundは*roundupと同じ値
        (そうでない場合は*rounddownと同じ値) */
```

【課題3-4】

[mainでの処理]

```
float f1; int r1, ru1, rd1;
f1 = 10.4;
round_value(f1, &ru1, &rd1, &r1);
printf("roundup: %d, rounddown: %d, round: %d\n",
       ru1, rd1, r1);
f1 = 10.5;
round_value(f1, &ru1, &rd1, &r1);
printf("roundup: %d, rounddown: %d, round: %d\n",
       ru1, rd1, r1);
f1 = 10.0;
round_value(f1, &ru1, &rd1, &r1);
printf("roundup: %d, rounddown: %d, round: %d\n",
       ru1, rd1, r1);
```

[実行結果]

```
roundup: 11, rounddown: 10, round: 10
roundup: 11, rounddown: 10, round: 11
roundup: 10, rounddown: 10, round: 10
```

まだ余裕のある人は…

【課題3-5】

仮引数で与えられた3つのchar型の文字から構成される3桁の整数値を求める関数to_integer()を作成してください。

[この関数のプロトタイプ宣言]

```
void to_integer(int *result, char *ch1, char *ch2,
                char *ch3);

 /*  *ch1が「100 の位」、*ch2が「10 の位」、
       *ch3が「1 の位」となる値を求め、*result に代入する */
 /* '0'～'9'の文字は'0'との差分を求めることで、
       整数値にすることができる */
 /* *ch1, *ch2, *ch3は'0'にリセットされる */
```

【課題3-5】

[mainでの処理]

```
int r;
char c1, c2, c3;
c1 = '2'; c2 = '6'; c3 = '7';
to_integer(&r, &c1, &c2, &c3);
printf("r: %d, c1: %c, c2: %c, c3: %c\n", r, c1, c2, c3);
```

[実行結果]

```
r: 267, c1: 0, c2: 0, c3: 0
```

小テストについて

小テストの注意点

- □ 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）

- □ 小テスト中は、演習室外へのネットワークアクセスは遮断される。

小テストについて

小テスト中に参照できるもの

□ **教科書, 配付資料**

□ **自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル**

□ **小テストでは紙媒体のものは参照可能**

□ **上記以外の情報を参照することは不正行為とする**
例：USBで接続された機器に保存されているファイルの参照
ネットワークを介した情報の参照、など