

# プログラミング基礎

<http://bit.ly/prog2d>

## 演算子 (1)

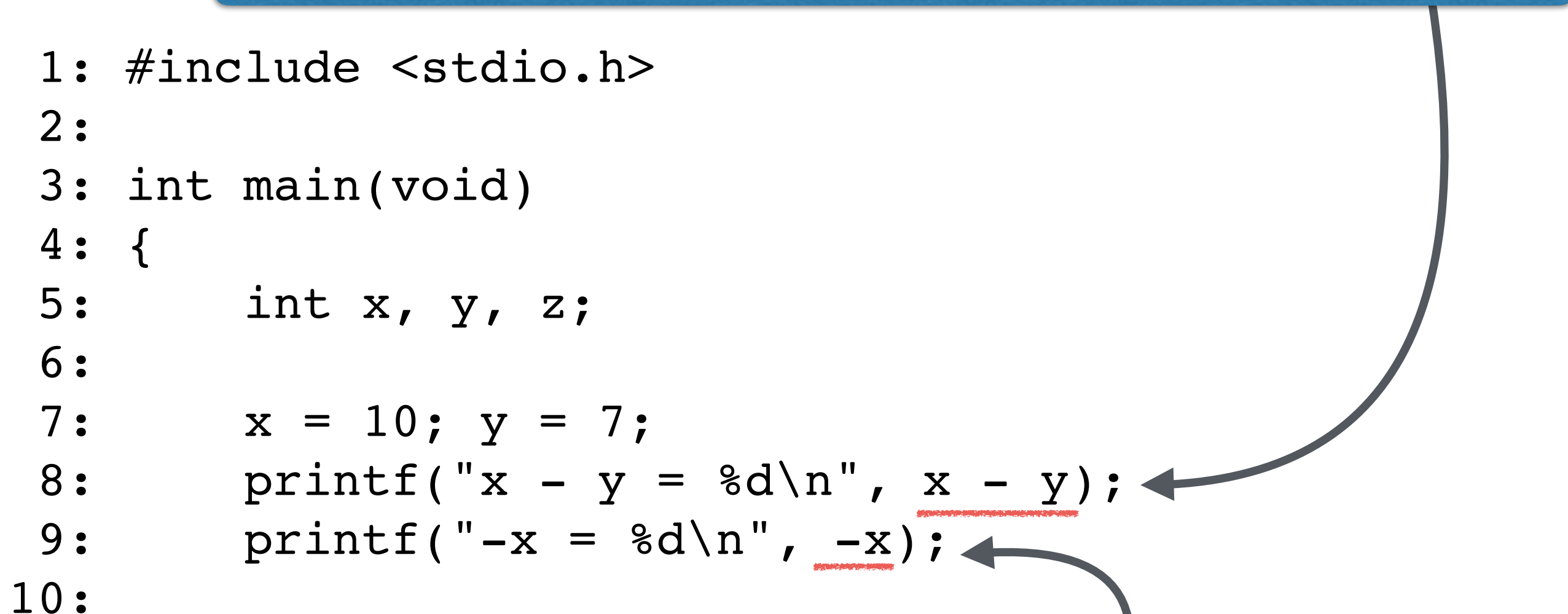
基本的な演算

前期 第13週

2017/7/17

【Point 1】 式は、演算子とオペランドから構成されている。  
例えば「x - y」という式は、「-」が**演算子**、「x」「y」が  
**オペランド**となる。(p.82 表4-1)

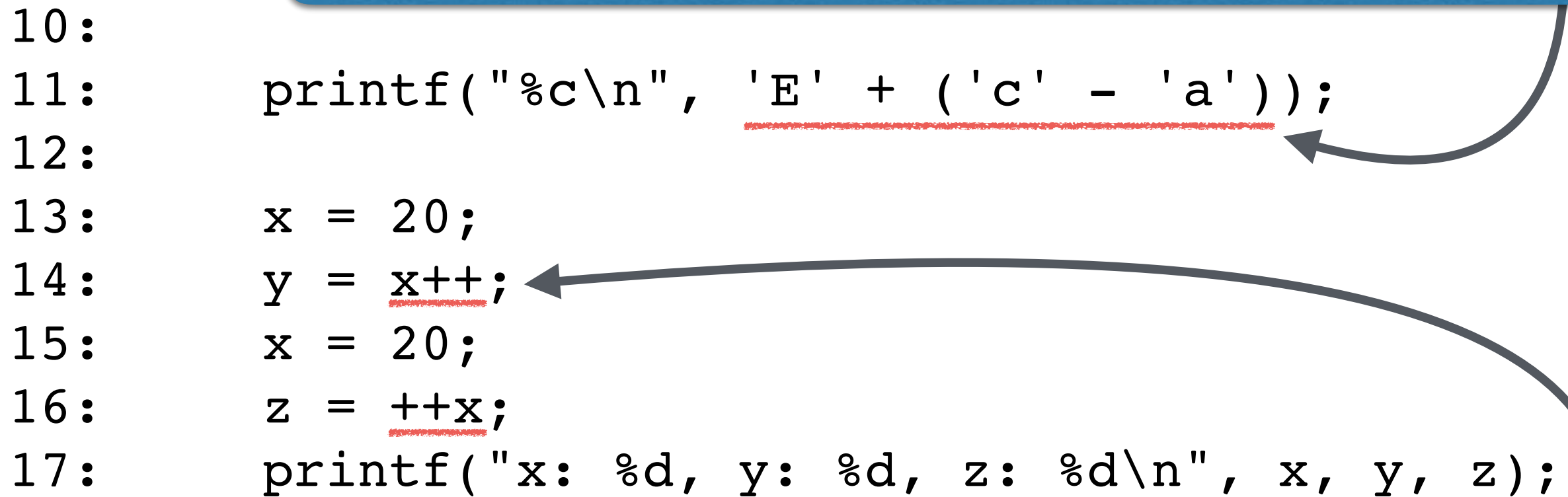
```
1: #include <stdio.h>
2:
3: int main(void)
4: {
5:     int x, y, z;
6:
7:     x = 10; y = 7;
8:     printf("x - y = %d\n", x - y);
9:     printf("-x = %d\n", -x);
10:
```



【Point 2】 オペランドを2個必要とする演算子のほかに、  
1個だけ必要とする演算子もあり、特にこれを**単項演算子**  
と呼ぶ。例えば「-10」は「負数」を表す単項演算子「-」  
を使っている。(p.83)

【Point 3】 char型の値に対しても、**整数と同じ演算子**を使える。（文字コードとして演算する）  
「'c' - 'a'」は整数の2となり、この2が 'E' に加算される。


```
10:
11:     printf("%c\n", 'E' + ('c' - 'a'));
12:
13:     x = 20;
14:     y = x++;
15:     x = 20;
16:     z = ++x;
17:     printf("x: %d, y: %d, z: %d\n", x, y, z);
```



【Point 4】 「++」を**インクリメント演算子**、「--」を**デクリメント演算子**という。例えば、「x++」は「x=x+1」と同じ意味になる。この演算子は、オペランドの前後どちらにも置ける（前置/後置）が、**処理の順番が異なる**。（前置はインクリメントをしてから代入するが、後置は代入してからインクリメントする）（p.84～87）

【Point 5】 代入「=」の前に演算子を組み合わせた「演算と代入を一度に行う」演算子のことを**代入演算子**という。例えば、「a += b」は「a = a + b」と同じ意味になる。(p.88～p.89)  
「x %= 10000」は「x = x % 10000」と同じ意味になる。

```
18:
19:     x = 39800;
20:     x %= 10000;
21:     printf("%d\n", x);
22:
23:     return 0;
24: }
```



# 【練習13-1】

サンプルプログラムをコンパイルして、  
実行結果を確認しましょう。

# 【課題13-1】

`scanf()`で入力された整数 $x$ に対して、日本円とした時の紙幣・硬貨の内訳を求めるプログラムを`main()`の中に作成してください。

ただし、大きい金額の紙幣・硬貨の方が優先されるようにしてください。（つまり、10000円の内訳は「1円硬貨10000枚」ではなく「10000円紙幣1枚」の方が優先される）

- ▶ まず、 $x$ と10000の商を出力し、 $x$ と10000の余りを求め、その余りと5000の商を出力し…、を1円硬貨まで続ける。
- ▶ （できる人は…）紙幣と硬貨の種類を配列に入れて、forの繰り返し処理で作るとコードが少なくなる。

# 【課題13-1】

[ 実行結果 (39827円の内訳の場合) ]

```
x > 39827
10000 yen: 3
 5000 yen: 1
 1000 yen: 4
  500 yen: 1
  100 yen: 3
   50 yen: 0
   10 yen: 2
    5 yen: 1
    1 yen: 2
```



# 【課題13-2】

引数chに対して、英小文字を大文字に変換する関数 `my_toupper()` を作成してください。  
ただし標準ライブラリ関数を使わずに作ってください。

[この関数のプロトタイプ宣言]

```
int my_toupper(int ch);
```

```
/* chが小文字（つまりa～zの範囲内）かどうかを調べる */
```

```
/* 小文字の場合、'a'と'A'の差分を、chから引くと大文字になる
```

```
（第11回の文字コード表参照） */
```

```
/* それ以外の場合は、chのままとする */
```

```
/* 大文字への変換にctype.hのtoupper()は使わない */
```



# 【課題13-2】

[mainでの処理]

```
printf("my_toupper: %c\n", my_toupper('b'));  
printf("my_toupper: %c\n", my_toupper('Q'));  
printf("my_toupper: %c\n", my_toupper('?'));
```

[実行結果]

```
my_toupper: B  
my_toupper: Q  
my_toupper: ?
```

# 【課題13-3】

グローバルに宣言された文字列strに対して、全て大文字に変換する関数my\_toupper\_str()を作成してください。

[この関数のプロトタイプ宣言]

```
void my_toupper_str();  
/* 文字列strの先頭から終端文字まで繰り返す処理を作り、  
   その繰り返し処理の中で、i番目の文字に対して、  
   関数my_toupper()を呼び出す */
```

# 【課題13-3】

[ 配列strをグローバル変数として宣言する ]

```
char str[20] = "Hello! :)";
```

[mainでの処理 (文字列strが "Hello! :)" の場合) ]

```
printf("str: %s -> ", str);  
my_toupper_str();  
printf("%s\n", str);
```

[ 実行結果 ]

```
str: Hello! :) -> HELLO! :)
```

# 【課題13-4】

- ▶ 引数chに対して、英字ならば一つ後ろの文字にする関数increase\_alpha()を作成してください。
- ▶ さらに、配列strの文字列に対して処理をする関数increase\_alpha\_str()を作成してください。

# 【課題13-4】

[この関数のプロトタイプ宣言]

```
int increase_alpha(int ch);  
    /* まず、関数の戻り値の初期値はchとしておく */  
    /* a～yまたはA～Yの場合は戻り値をch+1とする */  
    /* zの場合は、戻り値をaとする */  
    /* Zの場合、戻り値をAとする */  
  
void increase_alpha_str();  
    /* 文字列strの先頭から終端文字まで繰り返す処理を作り、  
       そこで関数increase_alpha()を繰り返し呼び出す */
```

# 【課題13-4】

[ 配列strをグローバル変数として宣言する ]

```
char str[20] = "(a~z) is increased.";
```

[mainでの処理]

```
printf("increase_alpha: %c\n", increase_alpha('b'));  
printf("increase_alpha: %c\n", increase_alpha('Q'));  
printf("increase_alpha: %c\n", increase_alpha('?'));  
printf("str: %s -> ", str);  
increase_alpha_str();  
printf("%s\n", str);
```

[実行結果]

```
increase_alpha: c  
increase_alpha: R  
increase_alpha: ?  
str: (a~z) is increased. -> (b~a) jt jodsfbtfe.
```

# まだ余裕のある人は…【課題13-5】

2から引数numまでの間にある素数を求めて出力する関数show\_prime()を作成してください。

[この関数のプロトタイプ宣言]

```
void show_prime(int num);  
/* 2～numまでの整数が素数であることを繰り返して調べる */  
/* 値iが素数かどうかは、2～i-1に割り切れる整数があるかで決まる */  
/* 割り切れる整数がiのみの場合は、素数なのでiを出力する */
```



# 【課題13-5】

[mainでの処理]

```
show_prime(100);
```

[実行結果]

2

3

(中略)

89

97

# 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