Programming I 0x05

if文（2009.05.10）

塩澤秀和 http://vilab.org

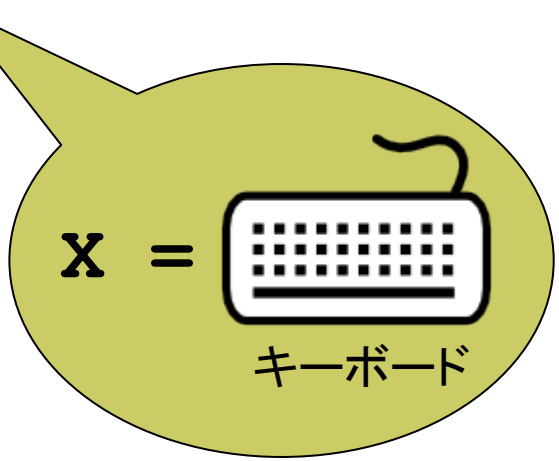
キーボードからの入力（復習）

```
#include <stdio.h>

int main(void)
{
    int x;
    int y;

    x = 18;
    y = x + 4;

    printf("答え:%d¥n", y);

    return 0;
}
```

実行するとプログラムが停止するので
キーボードから何か整数を打ち込んで
[Enter]キーを押す

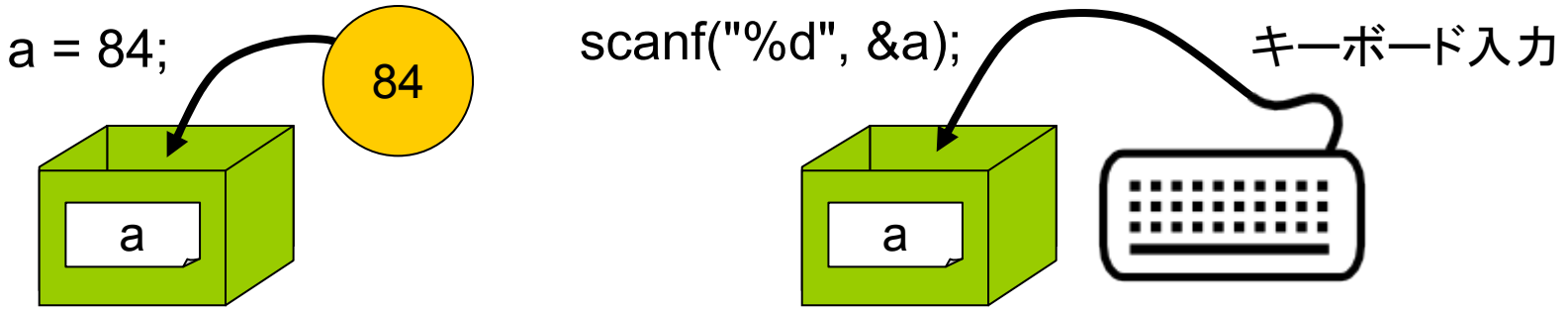
1行書き換えてみよう

```
scanf("%d", &x);
```



scanf 関数（復習）

- 標準入力からデータを読み込む
  - scanf("変換指定子", &変数);
  - 要注意!!「&」（変数の“アドレス”）が必要 ⇒ 意味はプロII



- データ型によって書式が違う
  - 整数（int）の入力： scanf("%d", &変数);
  - 実数（float）の入力： scanf("%f", &変数);
  - 実数（double）の入力：scanf("%lf", &変数); ←lは“エル”
  - 1文字（char）の入力： scanf("%c", &変数);

データ型の変換（復習）

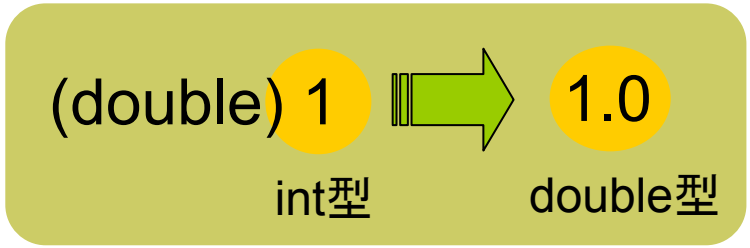
- 演算結果のデータ型
  - 整数型 と 整数型 ⇒ 整数型 （例：10 / 4 → 2）
  - 実数型 と 実数型 ⇒ 実数型 （例：10.0 / 4.0 → 2.5）
  - 整数型 と 実数型 ⇒ 実数型 （例：10 / 4.0 → 2.5）
  - 精度が高いほうのデータ型に暗黙的に変換される

- キャスト演算子（p.41）
  - データ型を明示的に変換する
    文法： (データ型名) 式
  - 例：整数→実数の変換（実数に直してから割り算）

```
heikin = (double)(a + b) / 2.0;    /* a, bはint型 */
```

  - 例：実数→整数の変換（小数点以下を切り捨てる）

```
discount = (int)(price * 0.8);
```



文字型について補足

- 文字(char)の正体
  - char型の正体は、1バイト整数の文字コード(文字番号)
  - ASCIIコードの環境では、'A'は整数の65とほぼ同じ意味
    ⇒ printf("%c", 'A'); と printf("%c", 65); は同じ表示結果

- char型入力の注意

× うまくいかない例

```
printf("a-> ");
scanf("%d", &a);
printf("c-> ");
scanf("%c", &c);
```

aの後の“改行文字”を
cに読み込んでしまう

○ うまくいく例

```
printf("a-> ");
scanf("%d", &a);
printf("c-> ");
scanf(" %c", &c);
```

書式中のスペースは、スペース、改行、
タブなどを読み飛ばす効果ある

```
printf("a c-> ");
scanf("%d %c", &a, &c);
```

```
printf("a-> ");
scanf("%d", &a);
printf("c-> ");
scanf(" ");
scanf("%c", &c);
```

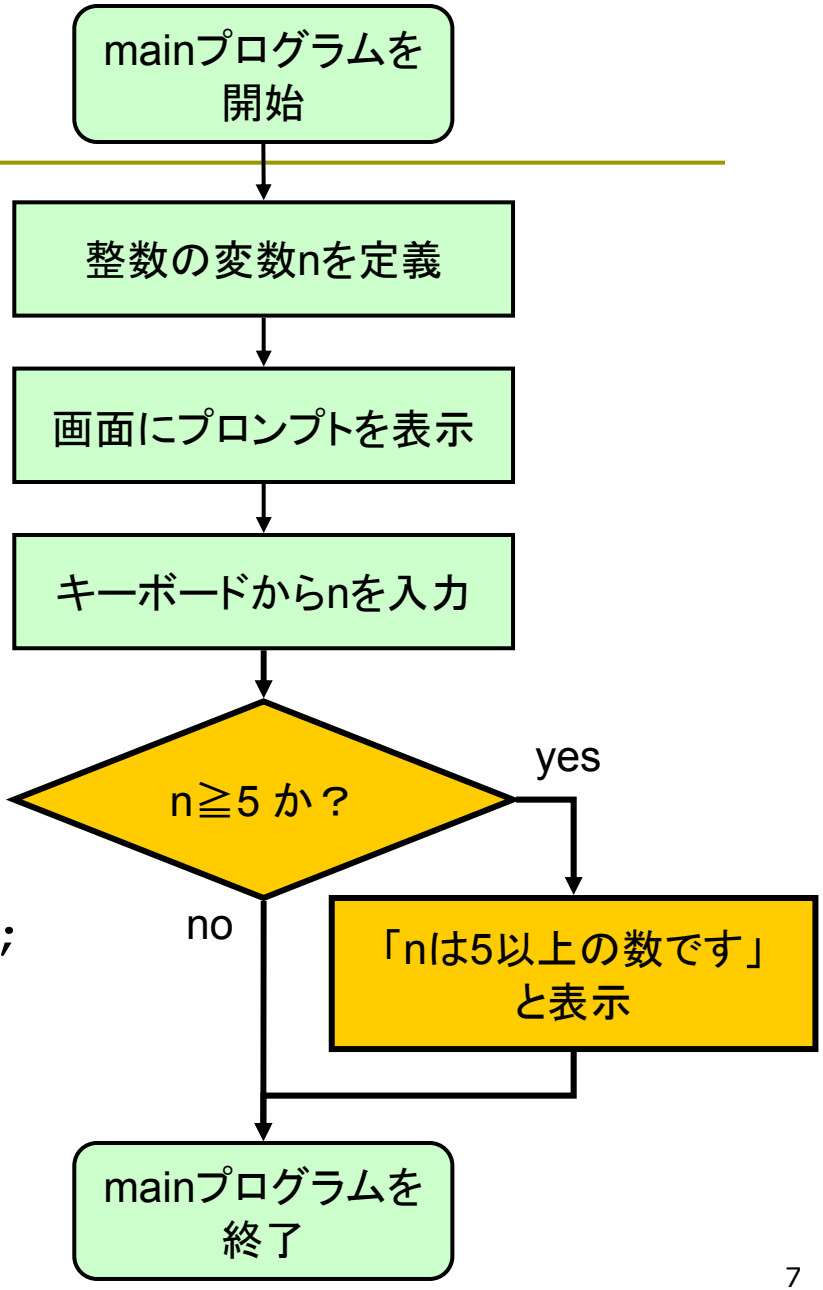
ASCIIコード表

16進 0x20-0x2f		16進 0x30-0x3f		16進 0x40-0x4f		16進 0x50-0x5f		16進 0x60-0x6f		16進 0x70-0x7f	
10進	文字	10進	文字	10進	文字	10進	文字	10進	文字	10進	文字
32	空白	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	¥	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

※ コード番号0～31（16進0x00～0x1f）と127（0x7f）は、改行・タブなどの制御文字

今日のプログラム

```
#include <stdio.h>

int main(void)
{
   int n;

   printf("整数 n -> ");
   scanf("%d", &n);

   if (n >= 5) {
      printf("nは5以上の数です¥n");
   }

   return 0;
}
```

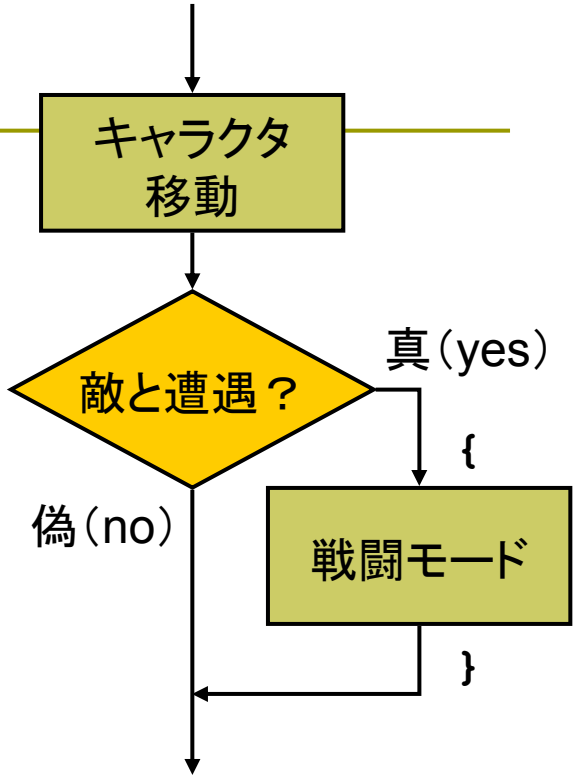


if文（p.44）

- if文（条件分岐）
  - ある条件があてはまったときだけ、決められた処理を実行する

```
if （条件式） {
    条件式が“真”のときの処理
    …
}
```

※ 処理が1文のときは { } を省略できる



ifのフローチャート

- “真”か“偽”か？
  - 真（true）　ある命題が正しい（条件が成立している）
  - 偽（false）　ある命題が正しくない（条件が不成立である）
  - C言語では、0で偽を、“0以外の数”（普通は1）で真を表す

関係演算子(p.45)

==は=と区別して、「イコール・イコール」と読むこともある

C言語	数学（読み方）	演算結果(意味)
`x == y`	＝（イコール）	xとyが等しければ真(1)、そうでなければ偽(0) 【要注意】C言語では「==」と「=」は違う意味！
`x != y`	≠（ノットイコール）	xとyが等しくなければ真(1)、等しければ偽(0)
`x < y`	＜（小なり）	xがyより小さければ真(1)、そうでなければ偽(0)
`x <= y`	≦（小なりイコール）	xがy以下ならば真(1)、そうでなければ偽(0)
`x > y`	＞（大なり）	xがyより大きければ真(1)、そうでなければ偽(0)
`x >= y`	≧（大なりイコール）	xがy以上ならば真(1)、そうでなければ偽(0)

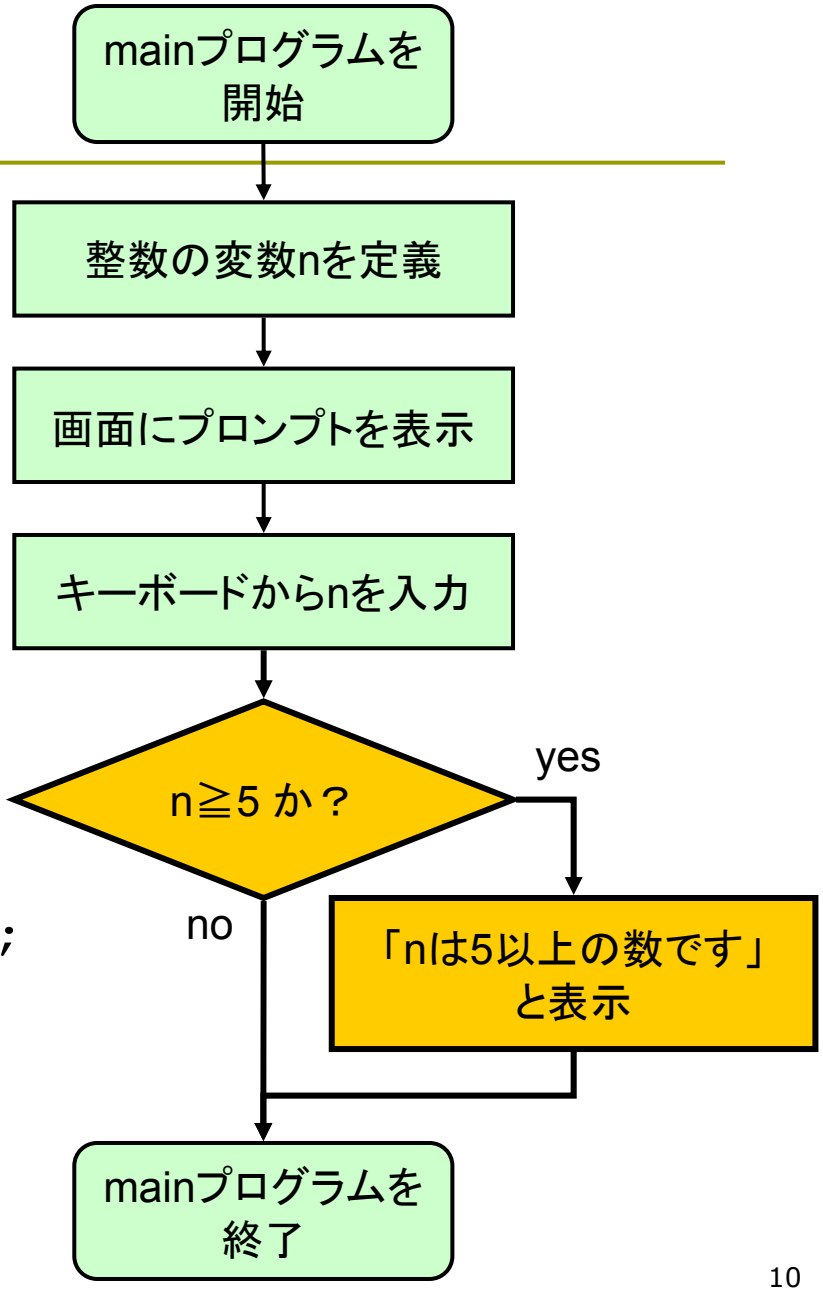
- それぞれどんな意味か？
  1. if (a == 10)
  2. if (x <= y)
  3. if (n != 0)
  4. if (i >= j)
  5. if (speed > limit)

条件式の値は真(1)か偽(0)

```
if (age < 20) {
  printf("お酒は飲めません¥n");
  printf("タバコも吸えません¥n");
}
```

if文の流れ

```
#include <stdio.h>

int main(void)
{
   int n;

   printf("整数 n -> ");
   scanf("%d", &n);

   if (n >= 5) {
      printf("nは5以上の数です¥n");
   }

   return 0;
}
```



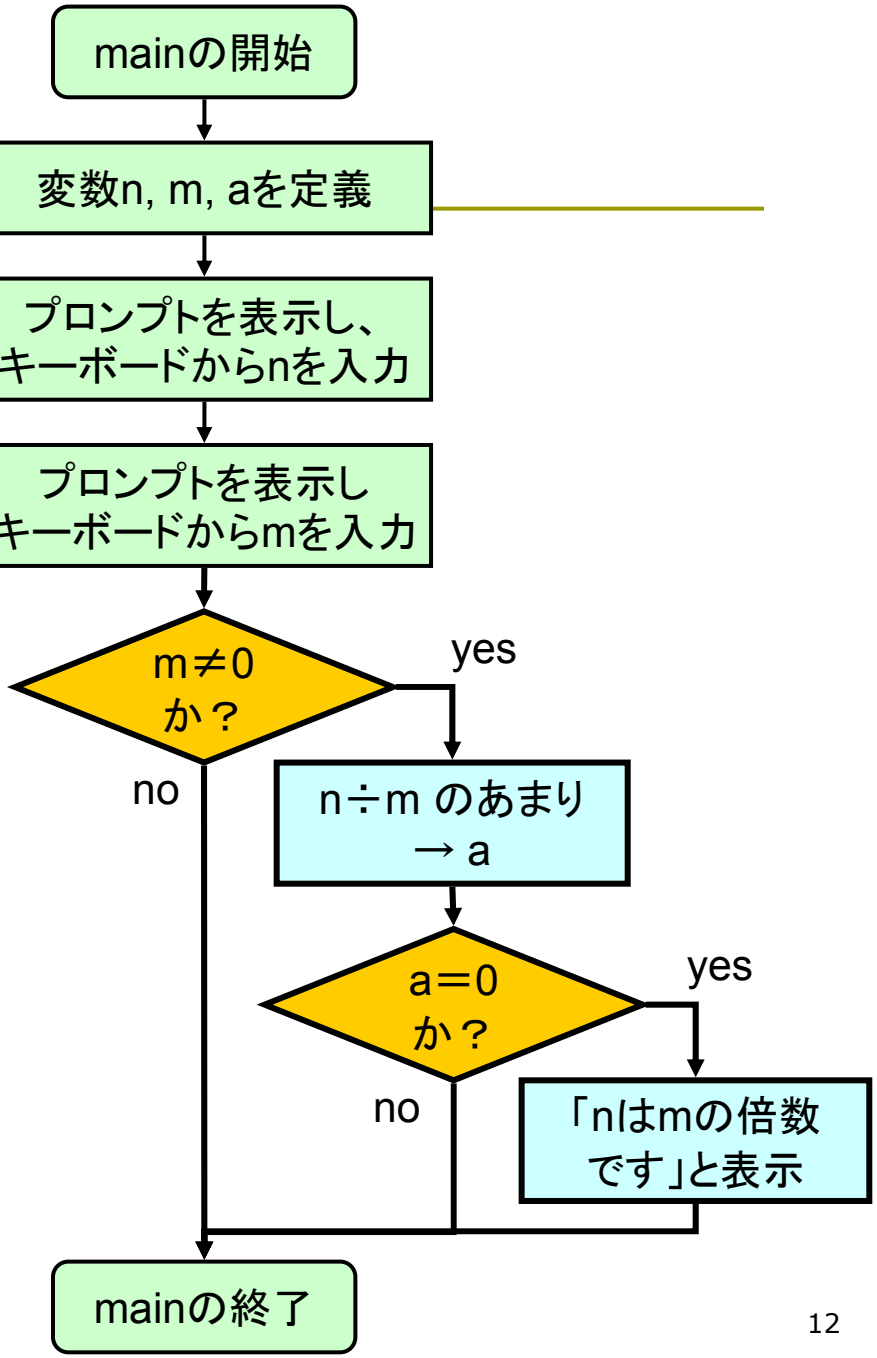
フローチャートを
書いてみよう

if文の使用例

```
#include <stdio.h>

int main(void)
{
  int n, amari;

  printf("整数を入力してね¥n");
  scanf("%d", &n);

  /* もし2で割り切れれば偶数 */
  amari = n % 2;
  if (amari == 0) {
    printf("%dは偶数！¥n", n);
  }

  printf("おしまい¥n");
  return 0;
}
```

if文の中にif文

```
#include <stdio.h>

int main(void)
{
  int n, m, a;

  printf("n="); scanf("%d", &n);
  printf("m="); scanf("%d", &m);

  /* m≠0じゃないと割り算できない */
  if (m != 0) {
    a = n % m;
    if (a == 0) {
      printf("nはmの倍数です¥n");
    }
  }
  return 0;
}
```



個数を数えるパターン

```
// キーボードから実数を3個入力し、
// 5.0以上の数値の個数を表示する
#include <stdio.h>

int main(void)
{
  int count = 0;
  double in;

  printf("実数3個？ ");

  // 1個目
  scanf("%lf", &in);
  if (in >= 5.0) {
    count = count + 1;
  }
  // 2個目
  scanf("%lf", &in);
  if (in >= 5.0) {
    count = count + 1;
  }

  // 3個目
  scanf("%lf", &in);
  if (in >= 5.0) {
    count = count + 1;
  }

  printf("5.0以上の数値は");
  printf("%d個です¥n", count);
  return 0;
}
```

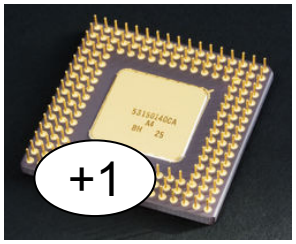
個数を0に初期化

あてはまれば個数に+1

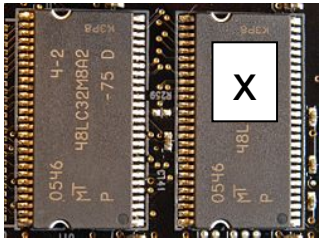
最後に個数を表示

「x = x + 1」?

- 意味:「xに1を加える」(x ← x + 1)
  - 両辺が等しくならない代入式
  - 動作のしくみ
    1. メモリ内のx → CPU内
    2. CPUで演算(1を加算)
    3. CPU内 → メモリ内のx

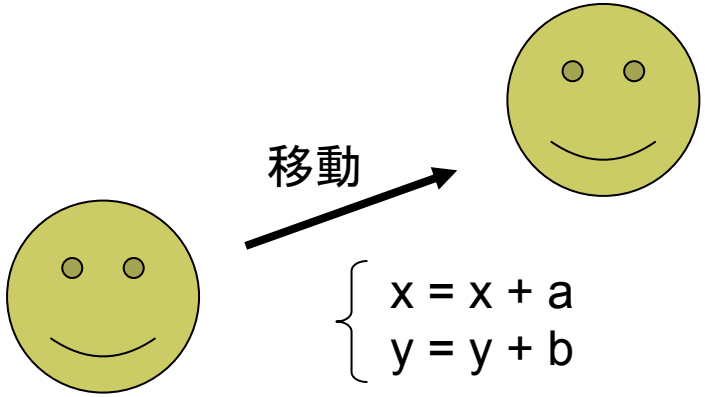


CPU(計算処理)



メモリ(データ格納)

- x = x + a はよく使うパターン
  - 個数や回数を数える
  - データの合計値の集計
  - 増えたり減ったりする点数
  - キャラクタの座標の移動



座標 (x, y)

ソースコードは見やすく！

- 上達の近道
  - すっきり統一した書式で書く
  - 単語や記号の区切りで、適当にスペースを入れる
  - 改行を入れて、関連する行のまとまりが分かるようにする
- インデント（字下げ）
  - ブロック { } に入るたびに、行頭を1段ずつ下げる
  - 下げ幅は、4文字が一般的
  - インデントの手動調整キー
    Tab, Shift+Tab, Backspace
  - メニュー→[編集]→[詳細]→[選択範囲のフォーマット]

- 流派はいろいろ

主流派

```
if (x > 0) {
    y = y + x;
}
printf("%f¥n", y);
```

```
if (x > 0)
{
    y = y + x;
}
printf("%f¥n", y);
```

```
if (x > 0)
  {
    y = y + x;
  }
printf ("%f¥n", y);
```

ソースコードの管理

- Visual Studioの問題点
  - 小さいプログラムをたくさん作るのに向かない
    - ソリューション → プロジェクト → ソース ファイル → ○○○.c
  - ある程度、プログラムを管理しやすくする方法
    ⇒ 1つの「ソリューション」の中に複数のプロジェクトを作る

- やりかた
  - [新しいプロジェクトの作成]で、[ソリューション]という欄があるので、ここを[ソリューションに追加]にする
  - そして、プロジェクトができたらプロジェクト名を右クリックし、[スタートアップ プロジェクトに設定]する
  - 実行したいプログラムが変わったら、そのたびにスタートアッププロジェクトを設定しなおすこと

演習問題

5a.「個数を数えるパターン」のプログラムのフローチャートを書きなさい。1行1行の意味を言葉で説明すること。

5b. 標準入力から整数を読み込み、それが0**未満**だった場合、「○○は負の数です」と表示するプログラムを作成しなさい。

5c. 整数 i, j をキーボードから読み込み、i と j が等しかったら、「等しいです」と表示するプログラムを作成しなさい。

5d. 下記の手順により、標準入力から実数を2つ読み込み、それらを小さい順にしてから表示するプログラムを作成しなさい。

1. double型の変数 a, b, c 定義し、aとbを読み込む
2. もし a > b だったら（if文）、aとbの値を（cを使って）交換する
3. 最後に、aとbの値を順に表示する

- 次回までの課題：教科書p.48～53 をよく読んでおく