プログラミング入門
第6回講義

ループ（その２）- for -

マークのあるサンプルプログラムは
`/home/course/prog0/public_html/2006/lec/source/`
下に置いてありますから、各自自分のディレクトリに
コピーして、コンパイル・実行してみてください

制御の流れ

- 前回の復習
  - if, switch の間違いやすいところ
- ループ
  - ループとは何か
  - ループの種類
  - while と for
- 式と演算子

良くあるミス(if編1)

- {}のつけ忘れ

```
if (a != b)
    a = b;
    printf("a,b were not equal\n");
```

このprintfはif文の外なので、必ず実行される！

- if文に「;」をつけてしまう

```
if (a != b);
    printf("a,b were not equal\n");
```

このprintfはif文の外なので、必ず実行される！

良くあるミス(if編2)

- ＝と＝＝の間違い

```
a = 1;
b = 0;
if (a = b){   /* bをaに代入し式の値もbとなる */
    printf("a,b were equal\n");
}
```

更にこの条件式は、bが
0の時：偽
0以外の時：真となる

- インデントをしよう！

```
if (a == b){    /* パターン1 */
    printf("a,b were equal\n");
}
if (a == b)     /* パターン2 */
{
    printf("a,b were equal\n");
}
```

自動的にインデントする方法は、
Lec13-7,8で説明します

良くあるミス(switch編１)

- switch文にbreakを付け忘れる

```
switch(a){
    case 1:
      printf("a=1¥n");
      /* break つけ忘れ */
    case 2:
      printf("a=2¥n");
      break;
    ..
}
```

aが１の時、２つとも表示される。
aが２の時は正常に動作する。

良くあるミス(switch編2)

- case文は定数だけ

```
switch(a){
    case >5:
      printf("a>5¥n");
      break;
    case 2:
      printf("a=2¥n");
      break;
    ..
}
```

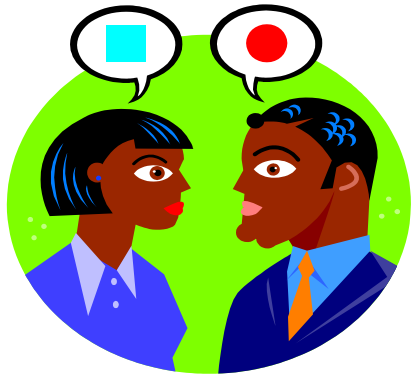
「case label does not reduce to an integer constant」
と言うメッセージが出てコンパイルエラーになる。

インクリメント, デクリメント演算子 (p.135)

- 変数に1加える（インクリメント）、
  または1減じる（デクリメント）
- インクリメント
  `i++, ++i` （iに対しては両方 `i=i+1`と同じ）
- デクリメント
  `i--, --i` （iに対しては両方 `i=i-1`と同じ）

◆前置と後置の違い(p.136)

- 前置(++i、--i)
  ⇒ iに1加え(減じ)てから、値を返す
- 後置(i++、i--)
  ⇒iの元の値を返してから、1加える(減じる)



例:i=6の時、実行後の値は

前置 `j = ++i;` ⇒ i:7 , j:7

後置 `j = i++;` ⇒ i:7 , j:6

- 単独で使用する場合は前置でも後置でも同じ

例:i=6の時、実行後の値は

前置 `++i;` ⇒ i:7

後置 `i++;` ⇒ i:7

/home/course/prog0/public_html/2006/lec/souce/lec06-1.c

代入演算子(p.87)

- =以外に, +=, -=, *=, /=, %=がある

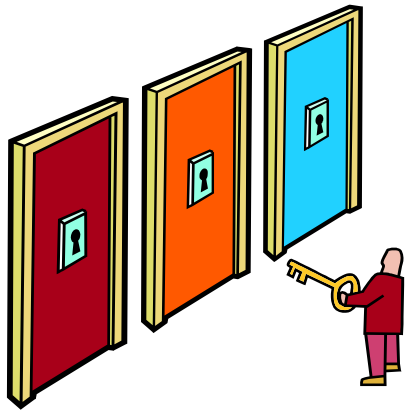
```
x += y  ⇒  x = x + y
x -= y  ⇒  x = x - y
x *= y  ⇒  x = x * y
x /= y  ⇒  x = x / y
x %= y  ⇒  x = x % y
```

ループとは何か(p.132)

- 同じことを、ある条件が成立している間何度も繰り返す
  - お金がある限り宝くじを買う
  - グラウンドを20周する
- Cの繰り返しの命令(文)
  - `while, for, (do-while)`

ループの種類(p.132)

- 見張り方式
  - ある特定の条件を満たしている間は、処理を繰り返す（例：aが0より大きい間）
  - 一般に繰り返し回数はあらかじめ決まっていない

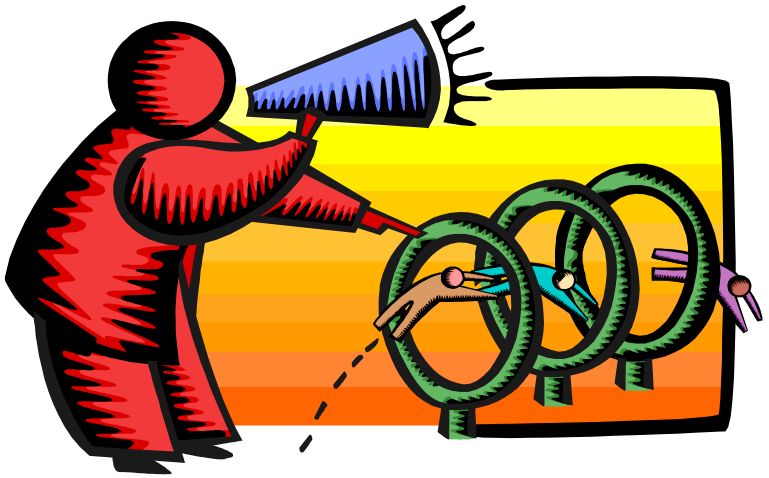


- カウンタ方式
  - ループの回数を数える変数（カウンタ変数などと呼ぶ）を用いて繰り返し処理を決定する（例：10回）
  - 一般的に繰り返し回数は固定

while 文（見張り方式に良く使われる）(p.132)

- 条件式が真の間はループの中身を実行し続け、条件式が偽になれば終了（ループの次の文を実行）する

```
while(条件式){
    文;
    ...
}
```



for 文（カウンタ方式に良く使われる）(p.143)

- おもにカウンタ方式のループに使用
- whileに、初期化とカウンタ変数の更新を追加

```
for(初期化；条件式；カウンタ変数更新){
    文;
    …
}
```

サンプルプログラム

```
#include <stdio.h>
main()
{
   int i,total;
   total = 0;
   for(i = 1 ; i <= 10 ; i++){
      total += i;
   }
   printf("1から10までの和は %d です¥n",total);
}
```

1から10までの和を求める

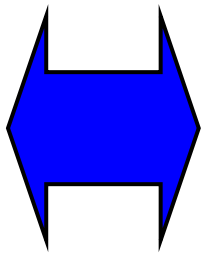
/home/course/prog0/public_html/2006/lec/source/lec06-3.c

forとwhileの比較(p.144)

```
for(初期化;条件式;カウンタ更新){
   文
   …
}
```

例

```
for(i = 0; i < 10; i++){
   sum += i;
}
```



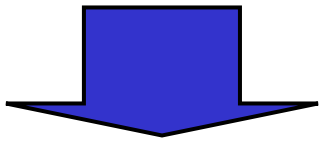
```
初期化
while(条件式){
   文
   …
   カウンタ更新
}
```

例

```
i = 0
while(i < 10){
   sum += i;
   i++;
}
```

カウンタ変数に関する注意

- カウンタ変数には通常整数型変数を用いる
- 浮動小数点数では、等号条件(==)が成立しないことがある（誤差を含むため）



- カウンタ変数には、整数型を用いたほうが無難

```
/home/course/prog0/public_html/2006/lec/source/lec06-4.c
```

いろいろなループ

```
for(i = 0 ; i < 10 ; i++ ){
   .....
}

for(i = 0 ; i <= 10 ; i += 2 ){
   .....
}

for(i = 1 ; i <= 16 ; i *= 2 ){
   .....
}
```

i : 0,1,...9のループ

i : 0,2,4...10のループ

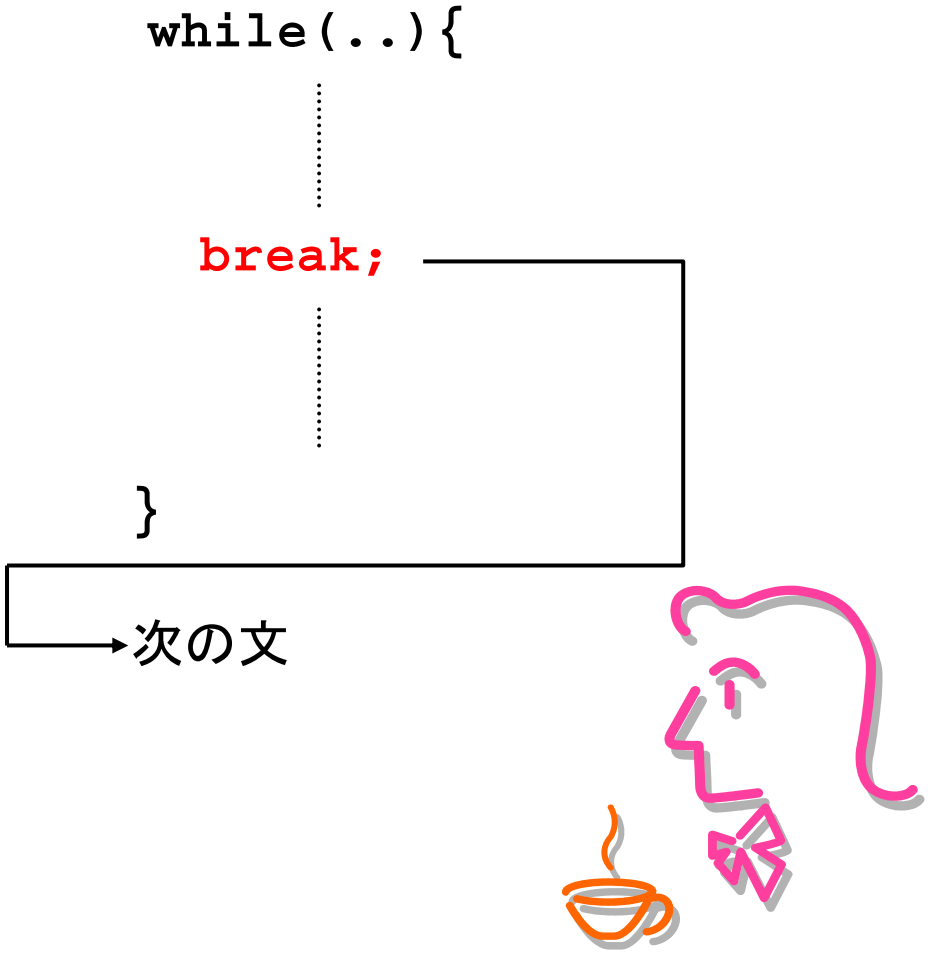
i : 1,2,4,8,16のループ

ループの制御(p.142,151)

- ループ実行中に、break, continueを使用してループの動作を変更する事が出来る
- ループ実行中に、ループの残りの全ての実行を取りやめる（ループからの脱出）: break
- ループ実行中にその回の残りの実行を取りやめ、次の回の実行に移る（ループの早送り）: continue
- while/for以外のループとしてdo-whileが、ループの制御としてその他にgotoがあるが、使用頻度も少ないのでプログラミング入門では割愛する（自習のこと）

break(p.146,152)

- ループ中のbreak以降を実行せず、ループの外に制御を移す
- 二重ループなど、入れ子になっているループの場合は、一番内側のループから抜け出るのみ



continue(p.142)

- ループ中のcontinue以降を実行せず、条件判断の直前に制御を移す
- ループの外には出ない
- forの場合はカウンタ更新を行い再度条件判断を行う

```
while(..){
        ⋮
    continue;
        ⋮
}
```

break/continueのプログラム例

```
#include <stdio.h>
main()
{
  int i, data;
  int num = 0, sum = 0;
  for(i = 0 ; i < 10 ; i++){
    scanf("%d",&data);
    if(data == 0) break;
    if(data < 0)  continue;
    sum += data;
    num++;
  }
  printf("有効データ数 : %d¥n",num);
  if(num != 0)
    printf("平均         : %f¥n",(float)sum/num);
}
```

データを10個読み正のデータのみの平均を計算するプログラム

データが0ならそれまで読んだデータと個数で平均を計算する

データがマイナスなら読み飛ばし平均の計算には使用しない（有効データ数にもカウントしない）

10個のデータのうち値が0より大きかった　「有効データ数」を表示

有効データ数が0でなければ平均を表示

/home/course/prog0/public_html/2006/lec/source/lec06-5.c

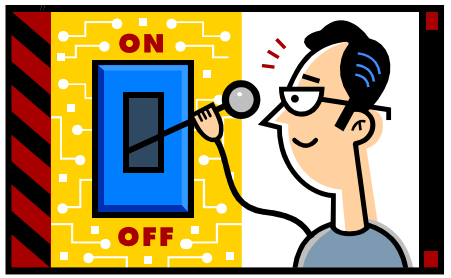
break/continueのプログラム実行例

```
std1dc1{s1000000}1: gcc lec06-5.c
std1dc1{s1000000}2: ./a.out
1 2 3 4 5 6 7 8 9 10 11 12
有効データ数：10
平均　　：5.500000
std1dc1{s1000000}3: ./a.out
1 -2 3 4 -5 0
有効データ数：3
平均　　：2.666667
std1dc1{s1000000}4:
```

1から10までの10個のデータが使用される

1、3、4の3個のデータが使用される

まとめ：if/switch-caseの使い分け



- switch-case文をつかう場面
  - 変数の値による場合分けで、しかも場合が比較的多い時（条件は等号でないといけない事に注意）
  - いくつかの場合の処理が同じなら、breakなしのcaseが使える
- if文をつかう場面
  - 条件が複雑な時（例えば、「条件AとBの真偽による4つの場合分け」にはswitch-caseを使用する事は非常に難しいだろう）
  - 条件が等号でない場合（！＝、＜、＝＜、＞、＝＞など）
  - switch-caseが使える時でも、あまり場合が多くない（3個ぐらい）だとif-elseを使用した方がコンパクトにまとめられる

まとめ：while/forの使い分け

- whileとforは書き換え可
  - だから最悪どちらかが書ければOK！
- while文をつかう場面
  - 見張り方式に良く使われる(Lec06-12)
  - ループする回数が分からないor分かり辛い場合
  - 無限ループにも良く使われる
- for文をつかう場面
  - カウンタ方式に良く使われる(Lec06-13)
  - ループする回数が定数や決まっている（例えば文字列長分ループなど）場合
  - for文は初期化、条件、更新を1文で書けるので以下の長所がある
    - コンパクトにまとめられる
    - 初期化等の書き忘れがない