情報システムプログラミングⅡ（17回目）

2024年10月9日（水）

3～4限

授業内容

- 講義内容（教科書の694～698ページ＋α）
  - ビット演算
  - Pythonにおけるビット演算
- 演習課題

ビット演算

■ビット単位で情報操作を行う演算子

- ビット単位の論理演算を行うビット論理演算子と，ビットの並びを右または左に動かす（シフトする）演算を行うシフト演算子がある

演算子	意味	使用例	解説
~	ビットごとの NOT	~a	a の各ビットの 0 と 1 を反転
&	ビットごとの AND	a & b	a と b のビット単位の AND ※1、※4
|	ビットごとの OR	a | b	a と b のビット単位の OR ※2、※4
^	ビットごとの XOR	a ^ b	a と b のビット単位の XOR ※3、※4
<<	左シフト	a << b	a を b ビット分、左へずらす
>>	右シフト	a >> b	a を b ビット分、右へずらす

※1　AND は、a と b が 1 なら結果は 1、それ以外は 0 を返す。
※2　OR は、a か b が 1 なら結果は 1、それ以外は 0 を返す。
※3　XOR は、a と b のビットが異なれば 1、等しければ 0 を返す。
※4　演算と同時に代入も行う &=、|=、^= 演算子も利用可能。

ビット演算

■シフト演算

- ビットの並びを動かす方向が，右だと右シフト演算，左だと左シフト演算
- 各ビットの扱いは場合によって異なる

シフト方式	溢れたビットの扱い	補充の扱い	
		左シフト時	右シフト時
論理シフト	捨てる	0 を補充	0 を補充
算術シフト	捨てる	0 を補充	符号ビットを補充
循環シフト	補充に使用	溢れたビット値	溢れたビット値

➢ 一般的に，符号無しは論理シフト，符号有りは算術シフト

ビット演算

■シフト演算の活用

- N個だけ右シフトすると2^{-N}倍，左シフトすると2^{N}倍になる
- 単に除算するよりも処理速度が速い

■ビット演算の活用（ビットフラグ）

- YES/NO（ON/OFF）などの2つの状態を1ビットで管理するビットフラグを効率よく利用できる

列挙型の利用
（文字列に値を設定）

```
enum {
  STATUS_POISON = 1,     // 2進数では00000001
  STATUS_SLEEP = 2,      // 2進数では00000010
  STATUS_SMALL = 4,      // 2進数では00000100
  STATUS_SILENT = 8      // 2進数では00001000
};
```

ビット演算

■ビット演算の活用（ビットフラグ）

```
printf("モンスターは毒状態になった！¥n");
m.status |= STATUS_POISON;

printf("モンスターは「眠り覚まし」を使った！¥n");
m.status &= ~STATUS_SLEEP;

printf("毒が効いてきた！¥n");
printf("（毒状態ならダメージを2だけ受ける）¥n");
if (m.status & STATUS_POISON) m.hp -= 2;

printf("「奇跡の石」を使った！¥n");
printf("（毒状態または眠りなら、HPを100回復）¥n");
if (m.status & (STATUS_POISON | STATUS_SLEEP)) m.hp = 100;
```

OR でフラグを立てる

NOT と AND でフラグを倒す

AND でフラグ判定

複数フラグの状況を一括して判定

ビット演算

■構造体におけるビットフィールド

- 構造体の各メンバのビット数を固定で指定できる
- メモリ領域の節約に繋がる

```
struct MonsterStatus {
  bool isPoison : 1,
  bool isSleep : 1,
  bool isSmall : 1,
  bool isSilent : 1
};
```

int a:2
double b:3
などと指定できる

Pythonにおけるビット演算

■ビット単位の論理演算

- AND：&，OR：|，XOR：^，NOT（反転）：~
- 単なる「論理演算」ではAND：and，OR：or

■シフト演算

- 右シフト演算：>>，左シフト演算：<<

```
x=10
y=11
print(bin(x))
print(bin(y))
print(bin(x&y))
```

bin関数により整数を
2進数に変換可能