ビット処理

ゲームソフト分野
1年　C++

ビット処理

ゲームの中で状態を管理するときフラグデータを使用する

例えば、それぞれの状態を管理する変数を宣言して、
変数を1か0かで状態を管理する方法がある

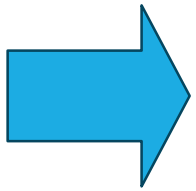
(例)キャラクターの状態

```
int IsPoison = 0;    //毒状態
int IsSleep = 0;     //眠り状態
int IsAtkUp = 0;     //攻撃力アップ
```

ビット処理

```
//何かしらの条件
{
    //毒状態
    IsPoison = 1;
}
//何かしらの条件
{
    //眠り状態
    IsSleep = 1;
}
```

```
//何かしらの条件
{
    //攻撃力Up状態
     IsAtkUp = 1;
}
```



それぞれの状態変化を
1つずつ個別の変数で
管理している

ビット処理

・状態を増やす際には**変数も増やしていく**必要がある…

```
int IsDefUp = 0;        //防御力アップ
int IsBurn = 0;         //火傷状態
...                     //どんどん増えていく…
```

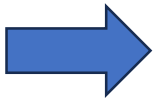
・また複数の状態を付与する場合
それぞれのフラグの値を一個ずつ変更しなければならない…

```
//毒状態になり攻撃力が下がる攻撃を受けた
IsPoison = 1;
IsAtkDown = 1;
```

ビット処理

- **ビット演算**を使うと
  1つの変数ですべての状態を表すことができる！

```
int IsPoison
int IsSleep
int IsAtkUp
    .....
```



```
unsigned int myState;
```

ステータスを管理する変数は1つにする

ビット処理

～考え方～
myStateは**32ビット(桁)の数値**
そのうちの1ビットにひとつの状態を当てはめる

```
unsigned int myState
```

```
0000 0000 0000 0000
0000 0000 0000 0000
```

0	0	0	0	0	0	0	0
防御力 Down	攻撃力 Down	防御力 Up	攻撃 Up	混乱	やけど	眠り	毒

ビット処理

～考え方～

(眠り&やけど&防御力Upの状態)

```
unsigned int myState;   //0010 0110
```

0	0	1	0	0	1	1	0
防御力 Down	攻撃力 Down	防御力 Up	攻撃 Up	混乱	やけど	眠り	毒

▶状態を変化させる場合は対応するビットを1にする

ビット処理

0	0	1	0	0	1	1	0
防御力 Down	攻撃力 Down	**防御力 Up**	攻撃 Up	混乱	**やけど**	**眠り**	毒

～必要な処理～

①**ビットを立てる**(例,毒の攻撃を受けたので毒のビットを**1にする**)

②**ビットを落とす**(例,毒消しが使われたので毒のビットを**0にする**)

③特定のビットが立っているかの**確認方法**

(例,**攻撃力Upしてたら～**,**防御力Downしてたら～**)

ビット演算

OR演算 |

どちらかが1なら1

```
  0
  1
-----
  1
```

AND演算 &

どちらもが1なら1

```
  0 1
  1 1
-----
  0 1
```

反転 ~

0なら1,1なら0

```
  0 1
-----
  1 0
```

列挙型enum

- 関連する定数をグループ化して、管理しやすくしたもの

- 文法
```
enum タグ名 { 定数1, 定数2, 定数3, …… };
```

- 例
```
enum Week {
   Sun, Mon, Tue, Wed, Thu, Fri, Sat
};
```

初期値を宣言しないと
```
   Sun:0 Mon:1 Tue:2 Wed:3 ··· Sat:6
```
という連番の整数値になる

列挙型enum

- 初期値を入れた例

```
enum Week {
   Sun, Mon, Tue = 10, Wed, Thu, Fri = 20, Sat
};
```

とすると、

```
Sun: 0 Mon: 1
Tue:10 Wed:11 Thu:12
Fri:20 Sat:21
```

のように設定した値以降が連番となる

状態管理 BitState

- 通常状態 Base = 0, //0000 0000(0)

- 毒 Poison = 1 << 0, //0000 0001(1)
- 眠り Sleep = 1 << 1, //0000 0010(2)
- 麻痺 Paralysis = 1 << 2, //0000 0100(4)
- 火傷 Burn = 1 << 3, //0000 1000(8)
- 攻撃↑ AtkUp = 1 << 4, //0001 0000(16)
- 攻撃↓ AtkDown = 1 << 5 //0010 0000(32)

通常状態に各状態を**OR演算**することで状態を変化させる

状態管理 BitState

- 通常状態 Base = 0,　　//0000 0000
- 毒　Poison = 1 << 0,　　//0000 0001
- 眠り Sleep = 1 << 1,　　//0000 0010

- 通常状態を毒状態にする場合

```
Status | Poison      0000 0000 (Status:Base)
                     0000 0001 (Poison)
        OR           0000 0001 (Status:Poison)
```

- さらに眠り状態にする場合

```
Status | Sleep       0000 0001 (Status:Poison)
                     0000 0010 (Sleep)
        OR           0000 0011 (Sleep+Poison)
```