## ［7］　制御構造（2）　くり返し，反復構造

### 1．for文　（所定回反復文）

for (式1 ; 式2 ; 式3)　　最初式1を1回だけ実行し，式2を評価する．
　文　　式2が成り立つなら，文を実行し，式3を実行する．
　　式2が満足されなくなるまで文を繰り返す．

例）
```
int i;
    for (i=1;i<3;i++)
       printf("Loop : %d\n",i);
```
実行結果）
```
Loop : 1
Loop : 2
```

例）次の場合，どのように表示されるか考えましょう．
```
int i, n=0;
for (i=1;i<3;i++){
   n+=i;
   printf("Loop : %d %d\n",i,n);
}
```

### 2．while文　（前判定反復文）

while　(式1)　式1が満足されなくなるまで文を繰り返す．
　文

例）
```
int i=0;                        整数変数iを初期化．
    while (i<3){                条件式 i<3 が成立しているか評価（判断）する．
      printf("Loop : %d\n",i);  iの値を表示．
      i++;                      iをインクリメント（1増やす）．
    }
```
実行結果）
```
Loop : 0
Loop : 1
Loop : 2
```

★ブロック｛　｝の使い方はif文と同じ．
☆for文，while文で繰り返す範囲（すなわち｛　｝の範囲）のことをforループ，whileループと言うことがある．

## 3．do while文　（後判定反復文）

```
do {                          文を１回実行した後，式1が満足されなくなるまで
  文                          文を繰り返す．
} while （式1）；
```

※while文との違いは，文が必ず１度は実行されること．以下を比べてみよう．

```
i=10;                         i=10;
do{                           while(i<1){
  文                            文
} while(i<1);                 };
```

☆for文とwhile文はどちらでも同様なプログラムがかける．

```
i=1;
while(i<3){                       for(i=1;i<3;i++){
   printf("Loop : %d\n",i);         printf("Loop : %d\n",i);
i++;                              }
}
```

※繰り返し回数がわかっている時はfor，そうでない場合はwhileを用いることが多い．