

C言語講習会

#6 繰り返し処理

1. 繰り返し処理

繰り返し処理とは

プログラミング言語では、あらかじめ決められた同じような処理を何度も繰り返すことがとあります。

このような処理を **繰り返し処理** といい、そのための構文を **繰り返し文(ループ文)** といいます。

(6)斜方投射

水平方向：速度 $v_x = v_0 \cos \theta$

変位 $x = v_0 \cos \theta \cdot t$

鉛直方向：速度 $v_y = v_0 \sin \theta - gt$

変位 $y = v_0 \sin \theta \cdot t - \frac{1}{2} gt^2$

引用：

https://w3e.kanazawa-it.ac.jp/e-scimath/contents/t22/textbook_t22_c02.pdf

1. 繰り返し処理

++ (インクリメント)による繰り返し処理

例えば、int型の **変数 i** を1ずつ増やしながら繰り返す方法が考えられます。

```
int i = 0;
```

```
i++;
```

```
printf(“%d回目の繰り返しです ¥n”, i );
```

```
⋮ n回繰り返し
```

```
i++;
```

```
printf(“%d回目の繰り返しです ¥n”, i );
```

1. 繰り返し処理

++ (インクリメント)による繰り返し処理

例えば、int型の **変数 i** を1ずつ増やしながら繰り返す方法が考えられます。

```
int i = 0;
```

```
i++;
```

```
printf(“%d回目の繰り返しです ¥n”, i );
```

```
∴ n回繰り返し
```

これをn行繰り返すのは、効率がわるい

```
i++;
```

```
printf(“%d回目の繰り返しです ¥n”, i );
```

1. 繰り返し処理

for文

効率のいい繰り返し処理の方法の1つに **for文** があります

```
int i;  
int n = 100;  
  
for(i = 0; i < n; i++){  
    ⋮ 100回繰り返し  
}
```

1. 繰り返し処理

for文

効率のいい繰り返し処理の方法の1つに **for文** があります

```
int i;  
int n = 10
```

繰り返した回数をカウントするための int型の変数 *i* を用意する。

```
for(i = 0; i < n; i++){  
    : 100回繰り返し  
}
```

1. 繰り返し処理

for文

効率のいい繰り返し処理の方法の1つに **for文** があります

```
int i;
```

```
int n = 100;
```

繰り返す回数を int型の **変数 n** に格納する。

```
for(i = 0; i < n; i++){
```

```
    ⋮ 100回繰り返し
```

```
}
```

1. 繰り返し処理

for文

効率のいい繰り返し処理の方法の1つに **for文** があります

```
int i;  
int n = 100;  
  
for(i = 0; i < n; i++)  
    : 100回繰り返し  
}
```

for 文

```
for( 変数の初期化 ; 条件式 ; 継続処理 ){ ... }
```

for文の()内の条件式を満たす間、{ }内のプログラムを処理する。

1. 繰り返し処理

for

効率

()内の内容

- ・最初にカウント用の **変数 i** に **0** を代入する
- ・ **$i < n$** が **真(ture)** なら、{ }内の処理を実行する
- ・ { }内の処理が終わったら、 **$i++$** でカウントを1増やす

上に戻る

in

in

(いずれ $i < n$ を満たさなくなるので、n回で繰り返しが終わる)

```
for(i = 0; i < n; i++){
```

```
    : 100回繰り返し
```

```
}
```

1. 繰り返し処理

for文

()内の ; (セミコロン)のつける位置に注意しましょう。

```
for(変数の初期化 ; 条件式 ; 継続処理 ){ ... }
```

最後は ; (セミコロン)をつけない

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

()文の中身で 変数i を宣言する

(C/C++ コンパイラのバージョンによっては、Warningになる可能性があります)

1. 繰り返し処理

for文

{ }内の処理が一つだけの場合、{ }を省略して一行で書くこともできます。

```
for(int i = 0; i < 10; i++) printf("%d回目 ¥n", i);
```

一行でfor文を書く際は、文末に ; (セミコロン) をつけることに注意

2. 条件付き繰り返し処理

continue文

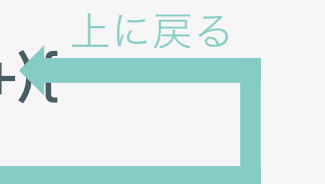
continue文が実行されると、即座にカウントが1増えて次の処理が始まります。

```
int i;  
  
for(i = 0; i < 5; i++){  
    if( i == 3 ) continue;  
    printf("%d回目 ", i);  
}  
printf("¥n"); // 改行
```

(実行結果)

0回目 1回目 2回目 4回目

```
for(i = 0; i < 5; i++){  
    continue;  
    printf("ここは実行されない¥n");
```



2. 条件付き繰り返し処理

break文

break文が実行されると、即座にfor文を終了して外に出ます。

```
int i;  
  
for(i = 0; i < 5; i++){  
    if( i == 3 ) break;  
    printf("%d回目 ", i);  
}  
printf("¥n"); // 改行
```

(実行結果)

3, 4回目は表示されない

0回目 1回目 2回目

```
for(i = 0; i < 5; i++){
```

```
    break;  
}
```

for文を終了して外に出る

3. 無限ループ

while文

for文の他にも繰り返し処理をする方法に **while文** があります

```
int i = 0;

while(i < 100){
    ∴ 100回繰り返し
    i++;
}
```

3. 無限ループ

while文

for文の他にも繰り返し処理をする方法に **while文** があります

```
int i = 0;
```

繰り返した回数をカウントするための int型の変数 i を用意する。(この時点で0を代入)

```
while(i < 100){
```

```
    ∴ 100回繰り返し
```

```
    i++;
```

```
}
```

3. 無限ループ

while文

for文の他にも繰り返し処理をする方法に **while文** があります

```
int i = 0;
while(i < 100){
    : 100回繰り返し
    i++;
}
```

while 文

```
while( 条件式 ){ ... }
```

while文の()内の条件式を満たす間、{ }内のプログラムを処理する。

while 【接続詞】 : ~する間

3. 無限ループ

while文

for文の他にも繰り返し処理をする方法に **while文** があります

```
int i = 0;
```

```
while(i < 100){
```

```
    ∴ 100回繰り返す
```

```
    i++;
```

```
}
```

for文とは違い、while文では {} 内で
カウント変数 i のインクリメントをおこなう

3. 無限ループ

while文

while文の最大の強みは、容易に **無限ループ** を実装できることです

```
int i = 0;

while(i == i){
    if( 終了条件 ) break;
    :
    : 繰り返し処理
}
```

3. 無限ループ

while文

while文の最大の強みは、容易に **無限ループ** を実装できることです

```
int i = 0;  
while(i == i){  
    if( 終了条件 )  
        : 繰り返し処理  
}
```

無限ループ

```
if( 終了条件 ) break;
```

無限ループも終了しなければ、他のプログラムが実行できず意味がない。

if文などを使って終了条件を必ず設置する

3. 無限ループ

while文

while文の最大の強みは、容易に **無限ループ** を実装できることです

```
int i = 0;
while(i == i){
    if( 終了条件 )
        : 繰り返し処理
}
```

終了条件

```
if( 終了条件 ) break;
```

無限ループも終了しなければ、他のプログラムが実行できず意味がない。

if文などを使って終了条件を必ず設置する

参考文献

- ・ 大川内隆郎, 大原竜男, *かんたん C言語* [改訂2版], 技術評論社, 2017.
- ・ 笈捷彦, 高田大二 他, *入門 C 言語*, 実教出版株式会社, 2019.