# 第6章 制御構造（ループ）

## 対応C規格

* **主要対象:** C90
* **学習内容:** for文、while文、do-while文、break文、continue文、ネストしたループ

## 学習目標

この章を完了すると、以下のことができるようになります：

* for文を使った繰り返し処理ができる
* while文とdo-while文の違いを理解する
* break文とcontinue文を適切に使える
* ネストしたループを理解して活用できる
* ループを使った実践的なプログラムを作成できる

## 概要と詳細

### 繰り返し処理とは？

プログラミングでは、同じ処理を何度も繰り返したい場面がたくさんあります。例えば：

* 1から100までの数を全て表示する
* 成績データを100人分処理する
* ゲームのメインループ（ユーザーが終了するまで続ける）
* ファイルのデータを最後まで読み込む

これらを一つ一つ手作業で書くのは大変です。そこで「ループ（繰り返し）」を使います！

#### なぜループが重要なの？

コンピュータの強みは「高速で正確な繰り返し処理」です。人間が苦手とする単調な繰り返し作業を、コンピュータは疲れることなく実行できます。この能力を活用するのがループです。

### for文

最も一般的なループ文で、初期化・条件・更新を一箇所で記述できます。

#### for文の仕組みを理解しよう

for文は「カウンター付きの繰り返し」と考えると分かりやすいです。例えば、ラジオ体操を10回やるとき：

1. 最初は1回目から始める（初期化）
2. 10回以下なら続ける（条件）
3. 1回終わったら回数を増やす（更新）

これをコードで表現すると：

for (int i = 1; i <= 10; i++) {  
 printf("%d回目のラジオ体操\n", i);  
}

#### 基本構文

for (初期化; 条件式; 更新式)   
 /\* 繰り返し実行される文 \*/

#### 基本的な使用例

#include <stdio.h>  
  
int main(void)  
{  
  
 int i;  
   
 /\* 1から10まで出力 \*/  
 for (i = 1; i <= 10; i++) {  
 printf("%d ", i);  
 }  
   
 printf("\n");  
   
 return 0;

#### for文の詳細動作

for (i = 0; i < 5; i++) {  
 printf("i = %d\n", i);  
}  
  
/\* 上記は以下と同等 \*/  
i = 0; /\* ① 初期化（1回のみ実行） \*/  
while (i < 5) { /\* ② 条件チェック \*/  
 printf("i = %d\n", i); /\* ③ ループ本体 \*/  
 i++; /\* ④ 更新式 \*/  
}

**実行の流れ：** 1. 初期化 → i = 0 2. 条件チェック → 0 < 5は真、続行 3. ループ本体実行 → “i = 0”を表示 4. 更新 → i = 1 5. 条件チェック → 1 < 5は真、続行 6. …これを繰り返し… 7. 更新 → i = 5 8. 条件チェック → 5 < 5は偽、ループ終了

#### さまざまなfor文のパターン

for文は非常に柔軟で、色々な書き方ができます：

/\* 逆順ループ（カウントダウン） \*/  
for (i = 10; i >= 1; i--) {  
 printf("%d ", i);  
}  
printf("発射！\n");  
  
/\* 2つずつ増加（偶数だけ表示） \*/  
for (i = 0; i <= 20; i += 2) {  
 printf("%d ", i);  
}  
  
/\* 複数変数の制御（両端から中央へ） \*/  
for (i = 0, j = 10; i < j; i++, j--) {  
 printf("i=%d, j=%d\n", i, j);  
}  
  
/\* 無限ループ（条件式を省略） \*/  
for (;;) {  
 /\* Ctrl+Cで止めるまで続く \*/  
 printf("無限ループ中...\n");  
}

**初心者のためのヒント：** - 最初は基本形（for (i = 0; i < n; i++)）をマスターしましょう - 複雑なfor文は読みにくくなりがちなので、シンプルに保つことが大切です

#### さまざまなfor文のパターン  
  
```c

### while文

条件が真である間、繰り返し処理を実行します。

#### while文って何？

while文は「〜である間、繰り返す」という意味です。条件が満たされている限り、処理を続けます。

**日常生活での例：** - お風呂にお湯を入れる → 「満タンになるまで」続ける - 宿題をする → 「全部終わるまで」続ける - 電車を待つ → 「電車が来るまで」待つ

これらは全て「ある条件が満たされるまで続ける」パターンです。

#### 基本構文

while (条件式)   
 /\* 繰り返し実行される文 \*/

#### while文の使用例

#include <stdio.h>  
  
int main(void)  
{  
  
 int count = 1;  
   
 while (count <= 5) {  
 printf("count = %d\n", count);  
 count++;  
 }  
   
   
 return 0;

#### while文の実践例

/\* ユーザー入力の処理 \*/  
int number;  
printf("正の数を入力してください（0で終了）: ");  
  
while (scanf("%d", &number) == 1 && number > 0) {  
 printf("入力された数: %d\n", number);  
 printf("次の数を入力してください（0で終了）: ");  
}  
  
  
printf("プログラムを終了します。\n");

**for文とwhile文の使い分け：** - **for文**：繰り返し回数が分かっている場合（例：10回繰り返す） - **while文**：条件で終了を判断する場合（例：ユーザーが0を入力するまで）

### do-while文   
  
最低1回は実行され、その後条件をチェックするループです。  
  
#### do-while文の特徴  
  
while文との大きな違いは「条件チェックのタイミング」です：  
  
- \*\*while文\*\*：最初に条件をチェック（条件が偽なら一度も実行されない）  
- \*\*do-while文\*\*：最後に条件をチェック（必ず一度は実行される）  
  
\*\*日常生活での例：\*\*  
  
レストランでの注文を考えてみましょう：  
1. メニューを見せる（必ず一度は見せる）  
2. 注文を聞く  
3. 「他に注文はありますか？」と聞く  
4. 「はい」なら1に戻る、「いいえ」なら終了  
  
これがdo-while文の考え方です！  
  
#### 基本構文  
  
```c  
do   
 /\* 最低1回は実行される文 \*/  
 while (条件式);

#### do-while文の使用例

#include <stdio.h>  
  
int main(void)  
{  
  
 int choice;  
   
 do {  
 printf("\nメニュー:\n");  
 printf("1. オプション1\n");  
 printf("2. オプション2\n");  
 printf("3. 終了\n");  
 printf("選択してください: ");  
 scanf("%d", &choice);  
   
 switch (choice) {  
 case 1:  
 printf("オプション1が選択されました\n");  
 break;  
 case 2:  
 printf("オプション2が選択されました\n");  
 break;  
 case 3:  
 printf("プログラムを終了します\n");  
 break;  
 default:  
 printf("無効な選択です\n");  
 break;  
 }  
 } while (choice != 3);  
   
 return 0;

### break文とcontinue文 ⏭

ループの流れを制御する特別な文です。

#### 途中でループを抜けたいとき

プログラムを書いていると、「ある条件を満たしたらループを抜けたい」という場面があります。そんなときに使うのがbreak文とcontinue文です。

**日常生活での例：** - **break**：探し物が見つかったら探すのをやめる - **continue**：嫌いな野菜は飛ばして次の料理を食べる

#### break文

ループを強制的に終了します。

#include <stdio.h>  
  
int main(void)  
{  
  
 int i;  
   
 for (i = 1; i <= 10; i++) {  
 if (i == 5) {  
 break; /\* i が 5 のときループを抜ける \*/  
 }  
 printf("%d ", i);  
 }  
 printf("\nループを抜けました\n");  
   
 return 0;  
  
/\* 出力: 1 2 3 4 \*/

#### continue文

現在の繰り返しをスキップして、次の繰り返しに進みます。

#include <stdio.h>  
  
int main(void)  
{  
  
 int i;  
   
 for (i = 1; i <= 10; i++) {  
 if (i % 2 == 0) {  
 continue; /\* 偶数の場合はスキップ \*/  
 }  
 printf("%d ", i);  
 }  
 printf("\n");  
   
 return 0;  
  
/\* 出力: 1 3 5 7 9 \*/

### ネストしたループ

ループの中にさらにループを含む構造です。

#### 二重ループを理解しよう

ネストしたループは「ループの中にループ」がある構造です。

**日常生活での例：**

学校の時間割を考えてみましょう： - 外側のループ：月曜日から金曜日まで（5回） - 内側のループ：1時間目から6時間目まで（6回） - 合計：5日 × 6時間 = 30コマ

これがネストしたループの考え方です！

#### 二重ループの例

#include <stdio.h>  
  
int main(void)  
{  
  
 int i, j;  
   
 /\* 九九表の作成 \*/  
 for (i = 1; i <= 9; i++) {  
 for (j = 1; j <= 9; j++) {  
 printf("%3d ", i \* j);  
 }  
 printf("\n");  
 }  
   
 return 0;

#### 三角形パターンの出力

#include <stdio.h>  
  
int main(void)  
{  
  
 int i, j;  
   
 /\* 星印の三角形 \*/  
 for (i = 1; i <= 5; i++) {  
 for (j = 1; j <= i; j++) {  
 printf("\* ");  
 }  
 printf("\n");  
 }  
   
 return 0;  
  
/\*  
出力:  
\*   
\* \*   
\* \* \*   
\* \* \* \*   
\* \* \* \* \*   
\*/

### ループでのbreak・continueの応用

#### ネストしたループでのbreak

#include <stdio.h>  
  
int main(void)  
{  
  
 int i, j;  
 int found = 0;  
   
 for (i = 1; i <= 10 && !found; i++) {  
 for (j = 1; j <= 10; j++) {  
 if (i \* j == 36) {  
 printf("発見: %d \* %d = 36\n", i, j);  
 found = 1;  
 break; /\* 内側のループを抜ける \*/  
 }  
 }  
 }  
   
 return 0;

#### ラベル付きbreak（goto文を使用）

#include <stdio.h>  
  
int main(void)  
{  
  
 int i, j;  
   
 for (i = 1; i <= 10; i++) {  
 for (j = 1; j <= 10; j++) {  
 if (i \* j == 36) {  
 printf("発見: %d \* %d = 36\n", i, j);  
 goto exit\_loops; /\* 両方のループを抜ける \*/  
 }  
 }  
 }  
   
exit\_loops:  
 printf("ループ終了\n");  
 return 0;

### 無限ループ ♾

意図的に終了しないループを作成することもあります。

#### 無限ループの作成方法

/\* 方法1: for文 \*/  
for (;;) {  
 /\* 無限ループ \*/  
 if (条件) {  
 break;  
 }  
}  
  
  
/\* 方法2: while文 \*/  
while (1) {  
 /\* 無限ループ \*/  
 if (条件) {  
 break;  
 }  
}  
  
  
/\* 方法3: do-while文 \*/  
do {  
 /\* 無限ループ \*/  
 if (条件) {  
 break;  
 }  
} while (1);

#### 無限ループの実用例

#include <stdio.h>  
  
int main(void)  
{  
  
 int choice;  
   
 while (1) { /\* 無限ループ \*/  
 printf("\n=== 計算機 ===\n");  
 printf("1. 足し算\n");  
 printf("2. 引き算\n");  
 printf("3. 終了\n");  
 printf("選択: ");  
   
 if (scanf("%d", &choice) != 1) {  
 printf("入力エラー\n");  
 break;  
 }  
   
 if (choice == 3) {  
 printf("終了します\n");  
 break;  
 }  
   
 /\* 計算処理... \*/  
 }  
   
 return 0;

### ループの最適化とベストプラクティス

#### 効率的なループ

/\* NG: 毎回strlen()を呼び出し \*/  
for (i = 0; i < strlen(str); i++) {  
 /\* 処理 \*/  
}  
  
  
/\* OK: 長さを事前に計算 \*/  
len = strlen(str);  
for (i = 0; i < len; i++) {  
 /\* 処理 \*/  
}

#### ループ変数の適切な使用

int main(void)  
{  
  
 int i; /\* C90では関数の先頭で宣言 \*/  
   
 for (i = 0; i < 10; i++) {  
 /\* iはループ外でも有効 \*/  
 }  
   
 printf("最終的なi = %d\n", i); /\* 10が出力 \*/  
   
 return 0;

## サンプルコード

### 基本的なループ例

プログラムファイル: examples/loop\_basic.c

for、while、do-while文の基本的な使用方法を学習します。

### ネストしたループ

プログラムファイル: examples/nested\_loops.c

二重・三重ループの使用例を学習します。

### break・continue文

プログラムファイル: examples/break\_continue.c

ループ制御文の使い方を学習します。

### コンパイルと実行

# examples ディレクトリに移動  
cd examples  
  
# C90準拠でコンパイル  
gcc -std=c90 -Wall -Wextra -pedantic loop\_basic.c -o loop\_basic  
  
# 実行  
./loop\_basic

## 演習課題

### 基礎問題

1. **数列の出力**
   * 1から20までの数を出力するプログラムを3種類のループで作成してください
2. **偶数・奇数の判定**
   * 1から50までの数で、偶数のみを出力するプログラムを作成してください
3. **階乗の計算**
   * 入力された数の階乗を計算するプログラムを作成してください

### 応用問題

1. **素数判定**
   * 入力された数が素数かどうかを判定するプログラムを作成してください
2. **フィボナッチ数列**
   * n項目までのフィボナッチ数列を出力するプログラムを作成してください
3. **パターン出力**
   * 以下のようなパターンを出力するプログラムを作成してください：

* \*  
  \*\*  
  \*\*\*  
  \*\*\*\*  
  \*\*\*\*\*  
  \*\*\*\*  
  \*\*\*  
  \*\*  
  \*

### 発展問題

1. **成績管理システム**
   * 複数の学生の成績を入力し、平均・最高・最低点を計算するプログラム
2. **簡易ゲーム**
   * 数当てゲームを作成してください（1-100の範囲で乱数を生成）
3. **データ検索**
   * 配列の中から特定の値を検索するプログラムを作成してください

## コンパイル方法

この章では以下のMakefileを使用してコンパイルができます：

# 全ての例題をコンパイル  
make all  
  
# 特定のプログラムをコンパイル  
make loop\_basic  
  
# 実行  
make run  
  
# クリーンアップ  
make clean

## 規格による違い

### C90での制限事項

* ループ変数は関数の先頭で宣言する必要がある
* for文内での変数宣言は不可

### C99以降の拡張

* for文内での変数宣言が可能
* for (int i = 0; i < 10; i++) /\* C99以降 \*/  
   /\* 処理 \*/

## よくある間違い

### 1. 無限ループの作成

/\* NG: 終了条件の記述ミス \*/  
for (i = 0; i > 10; i++) /\* 最初から条件が偽 \*/  
 /\* 実行されない \*/  
  
  
/\* NG: 更新式の記述ミス \*/  
for (i = 0; i < 10; i--) /\* 負の方向に進む \*/  
 /\* 無限ループ \*/

### 2. セミコロンの位置

/\* NG: for文の後にセミコロン \*/  
for (i = 0; i < 10; i++); /\* 空文が実行される \*/  
  
 printf("%d\n", i); /\* ループ外で実行 \*/  
  
  
/\* OK: 正しい記述 \*/  
for (i = 0; i < 10; i++)   
 printf("%d\n", i);

### 3. ループ変数の範囲外アクセス

/\* NG: 配列の範囲を超える \*/  
int arr[10];  
for (i = 0; i <= 10; i++) /\* i=10で範囲外 \*/  
 arr[i] = i;  
  
  
/\* OK: 正しい範囲 \*/  
for (i = 0; i < 10; i++)   
 arr[i] = i;

## 次の章へ

ループ処理を理解したら、[配列](../arrays/README.md) に進んでください。

## 参考資料

* [for文詳細](https://ja.cppreference.com/w/c/language/for)
* [while文詳細](https://ja.cppreference.com/w/c/language/while)
* [break・continue文](https://ja.cppreference.com/w/c/language/break)

# サンプルコード

## break\_continue.c

```c / break文とcontinue文の使い方  このプログラムは、ループ制御文である \* break文とcontinue文の使い方を示します。 \*/ #include <stdio.h>

int main(void) { int i, j; int sum; int num; int is\_prime; int count; int value; char ch;

/\* 1. break文の基本 \*/  
printf("=== break文の基本 ===\n");  
printf("1から10まで出力（5で中断）:\n");  
for (i = 1; i <= 10; i++)  
{  
 if (i == 5)  
 {  
 printf("\n5に到達したので終了します\n");  
 break;  
 }  
 printf("%d ", i);  
}  
printf("\n\n");  
  
/\* 2. continue文の基本 \*/  
printf("=== continue文の基本 ===\n");  
printf("1から10まで出力（偶数をスキップ）:\n");  
for (i = 1; i <= 10; i++)  
{  
 if (i % 2 == 0)  
 {  
 continue; /\* 偶数の場合は次の繰り返しへ \*/  
 }  
 printf("%d ", i);  
}  
printf("\n\n");  
  
/\* 3. 素数の判定（breakの活用） \*/  
printf("=== 素数判定 ===\n");  
printf("整数を入力してください: ");  
scanf("%d", &num);  
  
is\_prime = 1; /\* 素数フラグ（1:素数、0:素数でない） \*/  
  
if (num <= 1)  
{  
 is\_prime = 0;  
}  
else  
{  
 for (i = 2; i \* i <= num; i++)  
 {  
 if (num % i == 0)  
 {  
 is\_prime = 0;  
 break; /\* 約数が見つかったら終了 \*/  
 }  
 }  
}  
  
if (is\_prime)  
{  
 printf("%d は素数です\n", num);  
}  
else  
{  
 printf("%d は素数ではありません\n", num);  
}  
printf("\n");  
  
/\* 4. 負の数を除外した合計（continueの活用） \*/  
printf("=== 正の数の合計計算 ===\n");  
printf("5つの数を入力してください（負の数は無視されます）:\n");  
  
sum = 0;  
count = 0;  
for (i = 0; i < 5; i++)  
{  
 printf("%d番目の数: ", i + 1);  
 scanf("%d", &value);  
  
 if (value < 0)  
 {  
 printf(" → 負の数なのでスキップします\n");  
 continue;  
 }  
  
 sum += value;  
 count++;  
}  
  
printf("正の数の合計: %d（%d個）\n\n", sum, count);  
  
/\* 5. ネストしたループでのbreak \*/  
printf("=== ネストしたループでのbreak ===\n");  
printf("九九の表（5x5で中断）:\n");  
  
for (i = 1; i <= 9; i++)  
{  
 for (j = 1; j <= 9; j++)  
 {  
 if (i == 5 && j == 5)  
 {  
 printf("\n5x5で中断します\n");  
 break; /\* 内側のループのみ終了 \*/  
 }  
 printf("%2d ", i \* j);  
 }  
 if (i == 5 && j == 5)  
 {  
 break; /\* 外側のループも終了 \*/  
 }  
 printf("\n");  
}  
printf("\n");  
  
/\* 6. 無限ループとbreak \*/  
printf("=== 無限ループの制御 ===\n");  
printf("'q'を入力すると終了します:\n");  
while (1)  
{ /\* 無限ループ \*/  
 printf("文字を入力: ");  
 scanf(" %c", &ch);  
  
 if (ch == 'q' || ch == 'Q')  
 {  
 printf("終了します\n");  
 break;  
 }  
  
 printf("入力された文字: %c\n", ch);  
}  
printf("\n");  
  
/\* 7. 3の倍数と5の倍数をスキップ \*/  
printf("=== 3の倍数と5の倍数以外を表示 ===\n");  
printf("1から30まで（3の倍数と5の倍数を除く）:\n");  
  
for (i = 1; i <= 30; i++)  
{  
 if (i % 3 == 0 || i % 5 == 0)  
 {  
 continue;  
 }  
 printf("%d ", i);  
}  
printf("\n");  
  
return 0;

}```

## for\_loop\_basics.c

```c / for文の基本的な使い方  このプログラムは、for文を使った様々な \* ループパターンを示します。 \*/ #include <stdio.h>

int main(void) { int i, j; int sum; char ch; int n; int factorial;

/\* 1. 基本的なfor文：1から10まで出力 \*/  
printf("1から10まで: ");  
for (i = 1; i <= 10; i++)  
{  
 printf("%d ", i);  
}  
printf("\n\n");  
  
/\* 2. 逆順ループ：10から1まで出力 \*/  
printf("10から1まで（逆順）: ");  
for (i = 10; i >= 1; i--)  
{  
 printf("%d ", i);  
}  
printf("\n\n");  
  
/\* 3. 偶数だけを出力 \*/  
printf("0から20までの偶数: ");  
for (i = 0; i <= 20; i += 2)  
{  
 printf("%d ", i);  
}  
printf("\n\n");  
  
/\* 4. 累計の計算 \*/  
sum = 0;  
for (i = 1; i <= 100; i++)  
{  
 sum += i;  
}  
printf("1から100までの合計: %d\n\n", sum);  
  
/\* 5. 九九の表（ネストしたfor文） \*/  
printf("九九の表:\n");  
printf(" ");  
for (i = 1; i <= 9; i++)  
{  
 printf("%3d ", i);  
}  
printf("\n");  
printf(" ");  
for (i = 1; i <= 9; i++)  
{  
 printf("----");  
}  
printf("\n");  
  
for (i = 1; i <= 9; i++)  
{  
 printf("%2d |", i);  
 for (j = 1; j <= 9; j++)  
 {  
 printf("%3d ", i \* j);  
 }  
 printf("\n");  
}  
printf("\n");  
  
/\* 6. 文字の出力 \*/  
printf("アルファベット（大文字）: ");  
for (ch = 'A'; ch <= 'Z'; ch++)  
{  
 printf("%c ", ch);  
}  
printf("\n\n");  
  
/\* 7. 複数変数を使ったfor文 \*/  
printf("複数変数の例:\n");  
for (i = 0, j = 10; i < j; i++, j--)  
{  
 printf("i = %d, j = %d, 合計 = %d\n", i, j, i + j);  
}  
printf("\n");  
  
/\* 8. 階乗の計算 \*/  
n = 5;  
factorial = 1;  
printf("%d! = ", n);  
for (i = n; i >= 1; i--)  
{  
 factorial \*= i;  
 printf("%d", i);  
 if (i > 1)  
 printf(" x ");  
}  
printf(" = %d\n", factorial);  
  
return 0;

}```

## while\_dowhile.c

```c / while文とdo-while文の使い方  このプログラムは、while文とdo-while文の \* 基本的な使い方と違いを示します。 \*/ #include <stdio.h>

int main(void) { int i; int num; int sum; char choice; int target; int guess; int attempts;

/\* 1. 基本的なwhile文 \*/  
printf("=== while文の基本 ===\n");  
i = 1;  
while (i <= 5)  
{  
 printf("カウント: %d\n", i);  
 i++;  
}  
printf("\n");  
  
/\* 2. ユーザー入力を使ったwhile文 \*/  
printf("=== 合計計算（0で終了）===\n");  
sum = 0;  
printf("数値を入力してください（0で終了）: ");  
scanf("%d", &num);  
  
while (num != 0)  
{  
 sum += num;  
 printf("現在の合計: %d\n", sum);  
 printf("次の数値を入力してください（0で終了）: ");  
 scanf("%d", &num);  
}  
printf("最終合計: %d\n\n", sum);  
  
/\* 3. do-while文の基本 \*/  
printf("=== do-while文の基本 ===\n");  
i = 1;  
do  
{  
 printf("カウント: %d\n", i);  
 i++;  
} while (i <= 5);  
printf("\n");  
  
/\* 4. メニュー選択（do-whileの典型的な使用例） \*/  
printf("=== メニューシステム ===\n");  
do  
{  
 printf("\n--- メインメニュー ---\n");  
 printf("1. 挨拶\n");  
 printf("2. 現在時刻の表示（ダミー）\n");  
 printf("3. カウントダウン\n");  
 printf("Q. 終了\n");  
 printf("選択してください: ");  
 scanf(" %c", &choice);  
  
 switch (choice)  
 {  
 case '1':  
 printf("こんにちは！\n");  
 break;  
 case '2':  
 printf("現在時刻: 12:00（ダミー）\n");  
 break;  
 case '3':  
 printf("カウントダウン開始: ");  
 for (i = 5; i >= 1; i--)  
 {  
 printf("%d... ", i);  
 }  
 printf("発射！\n");  
 break;  
 case 'Q':  
 case 'q':  
 printf("プログラムを終了します。\n");  
 break;  
 default:  
 printf("無効な選択です。\n");  
 }  
} while (choice != 'Q' && choice != 'q');  
  
/\* 5. while vs do-whileの違い \*/  
printf("\n=== while vs do-while ===\n");  
  
/\* 条件が最初から偽の場合のwhile \*/  
printf("while文（条件が最初から偽）:\n");  
i = 10;  
while (i < 5)  
{  
 printf("これは実行されません\n");  
 i++;  
}  
printf("while文は一度も実行されませんでした\n");  
  
/\* 条件が最初から偽の場合のdo-while \*/  
printf("\ndo-while文（条件が最初から偽）:\n");  
i = 10;  
do  
{  
 printf("do-while文は最低1回実行されます（i = %d）\n", i);  
 i++;  
} while (i < 5);  
  
/\* 6. 数当てゲーム（簡易版） \*/  
printf("\n=== 数当てゲーム ===\n");  
target = 42; /\* 正解の数 \*/  
attempts = 0;  
  
printf("1から100までの数を当ててください!\n");  
  
do  
{  
 printf("予想を入力: ");  
 scanf("%d", &guess);  
 attempts++;  
  
 if (guess < target)  
 {  
 printf("もっと大きいです\n");  
 }  
 else if (guess > target)  
 {  
 printf("もっと小さいです\n");  
 }  
 else  
 {  
 printf("正解！ %d回で当たりました!\n", attempts);  
 }  
} while (guess != target);  
  
return 0;

}```