# 第2章 基本文法・Hello World - 演習問題

# 演習問題

## 基礎課題

### 課題2-1: 基本的な出力

プログラムを作成して以下を出力してください：

こんにちは、C言語の世界へ！  
私の名前は [あなたの名前] です。  
今日からプログラミングを始めます。

**要求事項:** - printf関数を使用すること - 適切なエスケープシーケンスを使用すること - [あなたの名前]の部分は実際の名前に置き換えること

**ファイル名:** ex2\_1\_hello\_intro.c

### 課題2-2: 変数と基本データ型

以下の変数を定義し、値を代入して出力するプログラムを作成してください：

**変数一覧:** - 整数型: 年齢（例：25） - 浮動小数点型: 身長（例：170.5） - 文字型: 血液型（例：‘A’） - 整数型: 好きな数字（例：7）

**出力例:**

=== 自己紹介データ ===  
年齢: 25歳  
身長: 170.5cm  
血液型: A型  
好きな数字: 7  
================

**要求事項:** - 適切なデータ型を選択すること - printfの書式指定子を正しく使用すること - コメントで各変数の説明を記述すること

**ファイル名:** ex2\_2\_personal\_data.c

### 課題2-3: 入力と出力

ユーザーから名前と年齢を入力してもらい、それを使って計算結果を表示するプログラムを作成してください。

**機能:** 1. 名前を入力してもらう 2. 年齢を入力してもらう 3. 10年後の年齢を計算する 4. 結果を表示する

**出力例:**

お名前を入力してください: 田中太郎  
年齢を入力してください: 25  
  
こんにちは、田中太郎さん！  
現在の年齢: 25歳  
10年後の年齢: 35歳  
プログラミングの学習、頑張ってください！

**要求事項:** - scanf関数を使用すること - 適切なバッファサイズを設定すること - 計算結果を変数に保存すること

**ファイル名:** ex2\_3\_age\_calculator.c

## 応用課題

### 課題2-4: 書式指定子の練習

様々な書式指定子を使って、数値を異なる形式で表示するプログラムを作成してください。

**表示する数値:** 123, 3.14159, 255

**出力例:**

=== 書式指定子のデモ ===  
整数 123 の表示:  
 10進数: 123  
 16進数: 7b  
 8進数: 173  
 フィールド幅5: | 123|  
 ゼロ埋め: |00123|  
  
実数 3.14159 の表示:  
 デフォルト: 3.14159  
 小数点以下2桁: 3.14  
 指数表記: 3.14159e+00  
 フィールド幅10.2: | 3.14|  
  
文字コード 255:  
 文字として: ÿ  
 16進数: ff  
 10進数: 255  
====================

**要求事項:** - 各種書式指定子を使用すること（%d, %x, %o, %f, %e, %c など） - フィールド幅とゼロ埋めを実演すること - 適切なコメントを記述すること

**ファイル名:** ex2\_4\_format\_demo.c

### 課題2-5: 簡単な計算機

四則演算を実行する簡単な計算機プログラムを作成してください。

**機能:** 1. 2つの数値を入力してもらう 2. 四則演算（+, -, \*, /）の結果を表示する 3. 割り算では整数除算と実数除算の両方を表示する

**出力例:**

簡単な計算機プログラム  
===================  
第1の数値を入力してください: 7  
第2の数値を入力してください: 2  
  
計算結果:  
 7 + 2 = 9  
 7 - 2 = 5  
 7 \* 2 = 14  
 7 / 2 = 3 (整数除算)  
 7 / 2 = 3.50 (実数除算)

**要求事項:** - 整数除算と実数除算を区別すること - ゼロ除算のチェックは不要（基礎課題のため） - 適切な変数名を使用すること

**ファイル名:** ex2\_5\_simple\_calculator.c

## 挑戦課題

### 課題2-6: 文字とASCIIコード

文字とASCIIコードの関係を学ぶプログラムを作成してください。

**機能:** 1. ユーザーから文字を入力してもらう 2. その文字のASCIIコードを表示する 3. ASCIIコード表の一部を表示する

**出力例:**

文字を1つ入力してください: A  
入力された文字: A  
ASCIIコード: 65  
  
ASCII表（32-126）の一部:  
 32: 33: ! 34: " 35: # 36: $ 37: % 38: & 39: '  
 40: ( 41: ) 42: \* 43: + 44: , 45: - 46: . 47: /  
 48: 0 49: 1 50: 2 51: 3 52: 4 53: 5 54: 6 55: 7  
 56: 8 57: 9 58: : 59: ; 60: < 61: = 62: > 63: ?  
 64: @ 65: A 66: B 67: C 68: D 69: E 70: F 71: G

**要求事項:** - getchar()またはscanf(" %c", &ch)を使用すること - ループを使ってASCII表を表示すること - 適切な書式で表を整列させること

**ファイル名:** ex2\_6\_ascii\_explorer.c

## 提出について

### ファイル構成

solutions/  
├── ex2\_1\_hello\_intro.c  
├── ex2\_2\_personal\_data.c  
├── ex2\_3\_age\_calculator.c  
├── ex2\_4\_format\_demo.c  
├── ex2\_5\_simple\_calculator.c  
└── ex2\_6\_ascii\_explorer.c

### コンパイルと実行

# コンパイル例  
gcc -o hello\_intro ex2\_1\_hello\_intro.c  
gcc -o personal\_data ex2\_2\_personal\_data.c  
gcc -o age\_calculator ex2\_3\_age\_calculator.c  
  
# 実行例  
./hello\_intro  
./personal\_data  
./age\_calculator

### 評価ポイント

1. **基本文法**: 正しいC言語の文法で記述されているか
2. **コーディング規約**: 適切なインデント、変数名、コメントが使用されているか
3. **機能性**: 要求された機能が正しく実装されているか
4. **入出力**: printf/scanfが適切に使用されているか
5. **書式指定**: 適切な書式指定子が使用されているか

### 学習のポイント

* C言語の基本的な入出力方法
* 変数の宣言と初期化
* 基本データ型の理解
* 書式指定子の使い方
* コメントの書き方
* プログラムの基本構造

頑張って取り組んでください！

# 解答例

## 解答例一覧

### 演習2-1: 自己紹介プログラム

* **ファイル**: <ex2_1_hello_intro.c>, <ex2_1_hello_intro_c99.c>
* **学習内容**: printf文の基本的な使い方、改行文字の使用
* **ポイント**: 複数行の出力とフォーマットの基本

### 演習2-2: 個人データ表示

* **ファイル**: <ex2_2_personal_data.c>, <ex2_2_personal_data_c99.c>
* **学習内容**: 変数の宣言と初期化、各種データ型の使用
* **ポイント**: int, float, char型の実用的な使い方

### 演習2-3: 年齢計算プログラム

* **ファイル**: <ex2_3_age_calculator.c>, <ex2_3_age_calculator_c99.c>
* **学習内容**: scanf関数による入力、計算処理
* **ポイント**: ユーザー入力を受け取って計算結果を表示

### 演習2-4: フォーマット指定子の実践

* **ファイル**: <ex2_4_format_demo.c>, <ex2_4_format_demo_c99.c>
* **学習内容**: printf文の書式指定子、桁数指定
* **ポイント**: %d, %f, %c, %sの実用的な使い方

### 演習2-5: 簡単な計算機

* **ファイル**: <ex2_5_simple_calculator.c>, <ex2_5_simple_calculator_c99.c>
* **学習内容**: 四則演算、浮動小数点数の処理
* **ポイント**: 複数の入力と計算結果の表示

### 演習2-6: ASCII文字探索

* **ファイル**: <ex2_6_ascii_explorer.c>, <ex2_6_ascii_explorer_c99.c>
* **学習内容**: 文字と数値の変換、ASCIIコードの理解
* **ポイント**: char型と整数型の関係性

## C90版とC99版の違い

### C90版（基本ファイル）

* すべての変数を関数の先頭で宣言
* /\* \*/ 形式のコメントを使用
* 従来のC言語の書き方に準拠

### C99版（\_c99.cファイル）

* 変数を使用する箇所の近くで宣言可能
* // 形式のコメントに対応
* より現代的なC言語の書き方

## コンパイルと実行

### C90版

gcc -Wall -Wextra -pedantic -std=c90 hello\_intro.c -o hello\_intro  
./hello\_intro

### C99版

gcc -Wall -Wextra -pedantic -std=c99 hello\_intro\_c99.c -o hello\_intro\_c99  
./hello\_intro\_c99

### Makefileを使用

make all # 全ての解答例をコンパイル  
make clean # 実行ファイルを削除  
make run-hello # hello\_introを実行

## 学習のポイント

1. **基本的な出力**: printf文の正しい使い方を習得
2. **変数の扱い**: 各種データ型の特徴を理解
3. **入力処理**: scanf関数の安全な使用方法
4. **書式指定**: 適切なフォーマット指定子の選択
5. **エラー処理**: 基本的な入力検証の考え方

## 注意事項

* すべての解答例は教育目的で作成されています
* 実際の開発では、より厳密なエラーハンドリングが必要です
* C90版を基本として学習し、C99版で現代的な書き方を確認してください

## 次のステップ

この章をマスターしたら、次は以下の章に進みましょう： - [第3章: データ型と演算子](../data-types/) - [第4章: 制御構造（条件分岐）](../control-if/) ## ex2\_1\_hello\_intro.c

/\*  
 \* 課題2-1の解答例: 基本的な出力  
 \* ファイル名: ex2\_1\_hello\_intro.c  
 \* 説明: printf関数とエスケープシーケンスの基本的な使用  
 \* 規格: C90準拠  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 /\* 基本的な文字列出力 \*/  
 printf("こんにちは、C言語の世界へ！\n");  
 printf("私の名前は 田中太郎 です。\n");  
 printf("今日からプログラミングを始めます。\n");  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. #include <stdio.h> - 標準入出力ライブラリの取り込み  
2. int main(void) - プログラムのエントリーポイント  
3. printf関数 - 文字列の出力  
4. \n - 改行を表すエスケープシーケンス  
5. return 0 - プログラムの正常終了を示す  
\*/

## ex2\_2\_personal\_data.c

/\*  
 \* 課題2-2の解答例: 変数と基本データ型  
 \* ファイル名: ex2\_2\_personal\_data.c  
 \* 説明: 基本データ型の変数宣言、初期化、書式指定子の使用  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 /\* 変数の宣言と初期化 \*/  
 int age = 25; /\* 年齢（整数） \*/  
 float height = 170.5f; /\* 身長（浮動小数点数） \*/  
 char blood\_type = 'A'; /\* 血液型（文字） \*/  
 int favorite\_number = 7; /\* 好きな数字（整数） \*/  
   
 /\* 自己紹介データの表示 \*/  
 printf("=== 自己紹介データ ===\n");  
 printf("年齢: %d歳\n", age);  
 printf("身長: %.1fcm\n", height);  
 printf("血液型: %c型\n", blood\_type);  
 printf("好きな数字: %d\n", favorite\_number);  
 printf("================\n");  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. 基本データ型の使い分け:  
 - int: 整数値  
 - float: 浮動小数点数（単精度）  
 - char: 1文字  
   
2. 変数の初期化:  
 - 宣言と同時に値を代入  
 - float型には f を付ける（170.5f）  
 - char型には シングルクォート（'A'）  
   
3. printf関数の書式指定子:  
 - %d: 整数（decimal）  
 - %f: 浮動小数点数（float）  
 - %c: 文字（character）  
 - %.1f: 小数点以下1桁表示  
   
4. コメントの書き方:  
 - スラッシュとアスタリスクで囲む（C90準拠）  
 - 変数の説明を記述  
\*/

## ex2\_3\_age\_calculator.c

/\*  
 \* 課題2-3の解答例: 入力と出力  
 \* ファイル名: ex2\_3\_age\_calculator.c  
 \* 説明: scanf関数を使った入力処理と計算  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 char name[50]; /\* 名前を格納する文字配列（50文字まで） \*/  
 int current\_age; /\* 現在の年齢 \*/  
 int future\_age; /\* 10年後の年齢 \*/  
   
 /\* ユーザーからの入力 \*/  
 printf("お名前を入力してください: ");  
 scanf("%49s", name); /\* 49文字まで読み込み（バッファオーバーフロー対策） \*/  
   
 printf("年齢を入力してください: ");  
 scanf("%d", &current\_age);  
   
 /\* 10年後の年齢を計算 \*/  
 future\_age = current\_age + 10;  
   
 /\* 結果の表示 \*/  
 printf("\n");  
 printf("こんにちは、%sさん！\n", name);  
 printf("現在の年齢: %d歳\n", current\_age);  
 printf("10年後の年齢: %d歳\n", future\_age);  
 printf("プログラミングの学習、頑張ってください！\n");  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. scanf関数の使用:  
 - %s: 文字列の入力  
 - %d: 整数の入力  
 - & (アドレス演算子): 変数のアドレスを指定  
   
2. 文字配列の宣言:  
 - char name[50]: 50文字まで格納可能  
 - scanf("%49s", name): バッファオーバーフロー対策  
   
3. 計算処理:  
 - future\_age = current\_age + 10  
 - 計算結果を変数に保存  
   
4. 改行文字の使用:  
 - \n で改行  
 - printf("\n") で空行挿入  
   
注意点:  
- scanf使用時はバッファサイズに注意  
- 実際のプログラムでは入力エラーのチェックも必要  
- 配列のサイズは格納する文字数+1（終端文字\0のため）  
\*/

## ex2\_4\_format\_demo.c

/\*  
 \* 課題2-4の解答例: 書式指定子の練習  
 \* ファイル名: ex2\_4\_format\_demo.c  
 \* 説明: printf関数の様々な書式指定子の使用方法  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 int integer\_num = 123;  
 double real\_num = 3.14159;  
 unsigned char char\_code = 255;  
   
 printf("=== 書式指定子のデモ ===\n");  
   
 /\* 整数の様々な表示形式 \*/  
 printf("整数 %d の表示:\n", integer\_num);  
 printf(" 10進数: %d\n", integer\_num);  
 printf(" 16進数: %x\n", integer\_num);  
 printf(" 8進数: %o\n", integer\_num);  
 printf(" フィールド幅5: |%5d|\n", integer\_num);  
 printf(" ゼロ埋め: |%05d|\n", integer\_num);  
 printf("\n");  
   
 /\* 実数の様々な表示形式 \*/  
 printf("実数 %.5f の表示:\n", real\_num);  
 printf(" デフォルト: %f\n", real\_num);  
 printf(" 小数点以下2桁: %.2f\n", real\_num);  
 printf(" 指数表記: %e\n", real\_num);  
 printf(" フィールド幅10.2: |%10.2f|\n", real\_num);  
 printf("\n");  
   
 /\* 文字コードの表示 \*/  
 printf("文字コード %d:\n", char\_code);  
 printf(" 文字として: %c\n", char\_code);  
 printf(" 16進数: %x\n", char\_code);  
 printf(" 10進数: %d\n", char\_code);  
   
 printf("====================\n");  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. 整数の書式指定子:  
 - %d: 10進数表示  
 - %x: 16進数表示（小文字）  
 - %X: 16進数表示（大文字）  
 - %o: 8進数表示  
   
2. フィールド幅指定:  
 - %5d: 5桁の幅で右寄せ  
 - %05d: 5桁の幅でゼロ埋め  
 - |%5d|: 表示範囲を見やすくするための記号  
   
3. 浮動小数点数の書式指定子:  
 - %f: 通常の小数表示  
 - %.2f: 小数点以下2桁まで表示  
 - %e: 指数表記（科学的記数法）  
 - %10.2f: 幅10、小数点以下2桁  
   
4. 文字の書式指定子:  
 - %c: 文字として表示  
 - %d: 文字コード（整数）として表示  
   
応用ポイント:  
- 複数の書式を組み合わせて使用可能  
- 表の整列や見やすい出力に活用  
- 数値の表示形式を用途に応じて選択  
\*/

## ex2\_5\_simple\_calculator.c

/\*  
 \* 課題2-5の解答例: 簡単な計算機  
 \* ファイル名: ex2\_5\_simple\_calculator.c  
 \* 説明: 四則演算の実装と整数除算・実数除算の違い  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 int num1, num2; /\* 計算に使用する2つの整数 \*/  
 int int\_result; /\* 整数演算の結果 \*/  
 double double\_result; /\* 実数演算の結果 \*/  
   
 /\* プログラムのタイトル表示 \*/  
 printf("簡単な計算機プログラム\n");  
 printf("===================\n");  
   
 /\* 数値の入力 \*/  
 printf("第1の数値を入力してください: ");  
 scanf("%d", &num1);  
   
 printf("第2の数値を入力してください: ");  
 scanf("%d", &num2);  
   
 printf("\n");  
   
 /\* 四則演算の実行と結果表示 \*/  
 printf("計算結果:\n");  
   
 /\* 加算 \*/  
 int\_result = num1 + num2;  
 printf(" %d + %d = %d\n", num1, num2, int\_result);  
   
 /\* 減算 \*/  
 int\_result = num1 - num2;  
 printf(" %d - %d = %d\n", num1, num2, int\_result);  
   
 /\* 乗算 \*/  
 int\_result = num1 \* num2;  
 printf(" %d \* %d = %d\n", num1, num2, int\_result);  
   
 /\* 除算（整数除算） \*/  
 int\_result = num1 / num2;  
 printf(" %d / %d = %d (整数除算)\n", num1, num2, int\_result);  
   
 /\* 除算（実数除算） \*/  
 double\_result = (double)num1 / num2; /\* キャストを使用 \*/  
 printf(" %d / %d = %.2f (実数除算)\n", num1, num2, double\_result);  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. 四則演算子:  
 - + : 加算  
 - - : 減算  
 - \* : 乗算（アスタリスク）  
 - / : 除算  
   
2. 整数除算と実数除算の違い:  
 - int / int → 整数除算（小数点以下切り捨て）  
 - (double)int / int → 実数除算（小数点以下あり）  
   
3. 型変換（キャスト）:  
 - (double)num1 → num1を一時的にdouble型に変換  
 - これにより実数除算が行われる  
   
4. 書式指定子:  
 - %d: 整数  
 - %.2f: 小数点以下2桁の実数  
   
5. プログラムの構成:  
 - 入力 → 処理 → 出力の基本的な流れ  
 - 見やすい出力のためのフォーマット  
   
注意点:  
- ゼロ除算のチェックは省略（基礎課題のため）  
- 実際のプログラムでは num2 が 0 でないかチェックが必要  
- キャストは演算の前に行う必要がある  
\*/

## ex2\_6\_ascii\_explorer.c

/\*  
 \* 課題2-6の解答例: 文字とASCIIコード  
 \* ファイル名: ex2\_6\_ascii\_explorer.c  
 \* 説明: 文字とASCIIコードの関係、ループによるASCII表の表示  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 char input\_char; /\* ユーザーが入力した文字 \*/  
 int ascii\_code; /\* ASCIIコード値 \*/  
 int i; /\* ループ用カウンタ \*/  
   
 /\* ユーザーからの文字入力 \*/  
 printf("文字を1つ入力してください: ");  
 scanf(" %c", &input\_char); /\* 空白文字をスキップするため先頭に空白 \*/  
   
 /\* 入力された文字のASCIIコードを表示 \*/  
 ascii\_code = (int)input\_char; /\* 文字を整数に変換 \*/  
 printf("入力された文字: %c\n", input\_char);  
 printf("ASCIIコード: %d\n", ascii\_code);  
 printf("\n");  
   
 /\* ASCII表の一部を表示 \*/  
 printf("ASCII表（32-126）の一部:\n");  
   
 /\* 32から126までのASCII文字を8文字ずつ表示 \*/  
 for (i = 32; i <= 126; i++) {  
 /\* 表示可能文字かチェック \*/  
 if (i == 32) {  
 printf("%3d: ", i); /\* スペース文字は特別に表示 \*/  
 } else {  
 printf("%3d: %c ", i, (char)i);  
 }  
   
 /\* 8文字ごとに改行 \*/  
 if ((i - 31) % 8 == 0) {  
 printf("\n");  
 }  
 }  
   
 /\* 追加情報の表示 \*/  
 printf("\n");  
 printf("ASCII表の説明:\n");  
 printf(" 0-31 : 制御文字（表示されない）\n");  
 printf(" 32 : スペース文字\n");  
 printf(" 33-126: 表示可能文字（英数字、記号など）\n");  
 printf(" 127 : DEL文字（制御文字）\n");  
   
 return 0;  
}  
  
/\*  
学習ポイント:  
1. 文字とASCIIコードの関係:  
 - 文字は内部的には数値（ASCIIコード）として扱われる  
 - (int)char\_var → 文字を整数に変換  
 - (char)int\_var → 整数を文字に変換  
   
2. scanf関数での文字入力:  
 - scanf(" %c", &var) → 先頭の空白で前の入力の改行文字をスキップ  
 - %c 指定子で1文字を読み込み  
   
3. forループの使用:  
 - for (i = 開始値; 条件; 増分) { 処理 }  
 - ASCII表の連続表示に活用  
   
4. 条件分岐:  
 - if文による特別な処理（スペース文字の場合）  
 - 改行タイミングの制御  
   
5. 書式指定子の応用:  
 - %3d: 3桁幅での整数表示  
 - 表を整列させるためのフォーマット  
   
応用ポイント:  
- ASCII表は文字コードの基礎  
- 文字と数値の相互変換は文字列処理の基本  
- ループを使った規則的な処理の例  
- 表形式での見やすい出力方法  
  
注意点:  
- C90では forループ内での変数宣言は不可  
- 変数 i は関数の先頭で宣言する必要がある  
\*/