C言語プログラミング教材 - 演習問題集（基礎編）

第1章〜第6章の演習問題

C言語学習教材プロジェクト

2024年

Table of Contents

[はじめに 6](#_Toc202360664)

[演習問題の構成 6](#_Toc202360665)

[学習の進め方 6](#_Toc202360666)

[第1章: 導入・環境構築 - 演習問題 7](#_Toc202360667)

[演習の目的 7](#_Toc202360668)

[基礎問題 7](#_Toc202360669)

[n 問題1-1: 環境構築確認 7](#_Toc202360670)

[n 問題1-2: コンパイラ情報の確認 8](#_Toc202360671)

[n 問題1-3: 基本的なプログラム作成 8](#_Toc202360672)

[応用問題 9](#_Toc202360673)

[n 問題1-4: 規格比較レポート 9](#_Toc202360674)

[n 問題1-5: コンパイル手順の詳細調査 9](#_Toc202360675)

[n 問題1-6: 環境固有情報の調査 10](#_Toc202360676)

[チャレンジ問題 10](#_Toc202360677)

[n 問題1-7: マルチ規格対応プログラム 10](#_Toc202360678)

[n 問題1-8: 自動環境診断ツール 11](#_Toc202360679)

[提出形式 12](#_Toc202360680)

[参考資料 12](#_Toc202360681)

[次の段階 12](#_Toc202360682)

[第2章: 基本文法・Hello World - 演習問題 13](#_Toc202360683)

[基礎問題 13](#_Toc202360684)

[n 問題2-1: 基本的な出力 13](#_Toc202360685)

[n 問題2-2: 変数と基本データ型 13](#_Toc202360686)

[n 問題2-3: 入力と出力 14](#_Toc202360687)

[応用問題 15](#_Toc202360688)

[n 問題2-4: 書式指定子の練習 15](#_Toc202360689)

[n 問題2-5: 簡単な計算機 16](#_Toc202360690)

[チャレンジ問題 17](#_Toc202360691)

[n 問題2-6: 文字とASCIIコード 17](#_Toc202360692)

[n 問題2-7: 文字列の基本操作 18](#_Toc202360693)

[n 問題2-8: プログラマー情報カード 19](#_Toc202360694)

[提出について 21](#_Toc202360695)

[n ファイル構成 21](#_Toc202360696)

[n コンパイルと実行 21](#_Toc202360697)

[n 評価ポイント 21](#_Toc202360698)

[n 学習のポイント 22](#_Toc202360699)

[第3章: データ型と変数 - 演習問題 23](#_Toc202360700)

[基礎問題 23](#_Toc202360701)

[n 演習3-1. 変数宣言と初期化 23](#_Toc202360702)

[n 演習3-2. 四則演算計算機 23](#_Toc202360703)

[n 演習3-3. データ型サイズの確認 24](#_Toc202360704)

[応用問題 25](#_Toc202360705)

[n 演習3-4. 型変換の理解 25](#_Toc202360706)

[n 演習3-5. スコープの実験 25](#_Toc202360707)

[n 演習3-6. 定数の活用 26](#_Toc202360708)

[チャレンジ問題 27](#_Toc202360709)

[n 演習3-7. 温度変換プログラム 27](#_Toc202360710)

[提出方法 27](#_Toc202360711)

[コンパイル例 28](#_Toc202360712)

[ヒント 28](#_Toc202360713)

[第4章: 演算子 - 演習問題 29](#_Toc202360714)

[基礎問題 29](#_Toc202360715)

[n 問題4-1: 四則演算計算機 29](#_Toc202360716)

[n 問題4-2: 比較と論理演算 29](#_Toc202360717)

[n 問題4-3: インクリメント・デクリメント 30](#_Toc202360718)

[応用問題 31](#_Toc202360719)

[n 問題4-4: ビット操作 31](#_Toc202360720)

[n 問題4-5: 条件演算子の活用 31](#_Toc202360721)

[n 問題4-6: 演算子優先順位 32](#_Toc202360722)

[チャレンジ問題 32](#_Toc202360723)

[n 問題4-7: 複合代入演算子の応用 32](#_Toc202360724)

[n 問題4-8: ビット演算による最適化 33](#_Toc202360725)

[n 問題4-9: 演算子組み合わせパズル 33](#_Toc202360726)

[提出方法 34](#_Toc202360727)

[コンパイル例 34](#_Toc202360728)

[評価ポイント 34](#_Toc202360729)

[ヒント 34](#_Toc202360730)

[第5章: 制御構造（条件分岐） - 演習問題 35](#_Toc202360731)

[演習の目的 35](#_Toc202360732)

[基礎問題 35](#_Toc202360733)

[n 問題5-1: 年齢による料金計算 35](#_Toc202360734)

[n 問題5-2: 電卓プログラム 36](#_Toc202360735)

[n 問題5-3: BMI判定プログラム 36](#_Toc202360736)

[応用問題 37](#_Toc202360737)

[n 問題5-4: うるう年判定 37](#_Toc202360738)

[n 問題5-5: 成績評価システム 38](#_Toc202360739)

[n 問題5-6: 曜日判定プログラム 39](#_Toc202360740)

[チャレンジ問題 40](#_Toc202360741)

[n 問題5-7: じゃんけんゲーム 40](#_Toc202360742)

[n 問題5-8: 交通信号シミュレーター 41](#_Toc202360743)

[n 問題5-9: 複雑な料金計算システム 41](#_Toc202360744)

[提出方法 42](#_Toc202360745)

[評価基準 43](#_Toc202360746)

[学習のポイント 43](#_Toc202360747)

[第6章: 制御構造（ループ） - 演習問題 44](#_Toc202360748)

[演習の目的 44](#_Toc202360749)

[基礎問題 44](#_Toc202360750)

[n 問題6-1: フィボナッチ数列 44](#_Toc202360751)

[n 問題6-2: 数当てゲーム 45](#_Toc202360752)

[n 問題6-3: 掛け算表 46](#_Toc202360753)

[応用問題 46](#_Toc202360754)

[n 問題6-4: 素数リスト 46](#_Toc202360755)

[n 問題6-5: 図形の描画 47](#_Toc202360756)

[n 問題6-6: 完全数の探索 48](#_Toc202360757)

[チャレンジ問題 49](#_Toc202360758)

[n 問題6-7: パスカルの三角形 49](#_Toc202360759)

[n 問題6-8: コンソールアニメーション 49](#_Toc202360760)

[n 問題6-9: 数字ピラミッド 50](#_Toc202360761)

[提出方法 51](#_Toc202360762)

[評価基準 51](#_Toc202360763)

[学習のポイント 52](#_Toc202360764)

[演習問題を解き終えたら 52](#_Toc202360765)

[チェックリスト 52](#_Toc202360766)

[次のステップ 52](#_Toc202360767)

[困ったときは 52](#_Toc202360768)

# はじめに

本書は、C言語プログラミング教材基礎編の演習問題集です。各章で学んだ内容を実践的に確認し、理解を深めるための問題を収録しています。

## 演習問題の構成

各章の演習問題は以下の構成になっています：

* **基礎問題**: 学習内容の基本的な理解を確認
* **応用問題**: 複数の概念を組み合わせた問題
* **発展問題**: より実践的な課題

## 学習の進め方

1. まず自分で問題を解いてみる
2. コンパイル・実行して動作を確認
3. 解答例と比較して理解を深める
4. 別解や改善点を考える

# 第1章: 導入・環境構築 - 演習問題

## 演習の目的

この演習を通して、C言語の開発環境が正しく構築されているかを確認し、基本的なコンパイル手順を習得します。

## 基礎問題

### 問題1-1: 環境構築確認

サンプルプログラム examples/environment\_check.c を使用して、以下の作業を実行してください。

1. **C90準拠でコンパイル・実行**

* make c90

1. **C99準拠でコンパイル・実行**

* make c99

1. **C11準拠でコンパイル・実行**

* make c11

1. **C17準拠でコンパイル・実行**

* make c17

1. **実行結果の確認**
   * 各規格でコンパイルしたプログラムの出力結果を比較してください
   * 規格による違いがあるかを確認してください

**期待される出力例:**

環境チェックプログラム  
==================  
コンパイラ: GCC  
C規格: C90  
ポインタサイズ: 8 bytes  
int型サイズ: 4 bytes  
long型サイズ: 8 bytes  
環境チェック完了

### 問題1-2: コンパイラ情報の確認

以下のコマンドを実行して、開発環境の情報を調べてください。

1. **コンパイラバージョンの確認**

* gcc --version
* **出力例:**
* gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0  
  Copyright (C) 2021 Free Software Foundation, Inc.

1. **対応規格の確認**

* gcc -std=c90 --help  
  gcc -std=c99 --help  
  gcc -std=c11 --help  
  gcc -std=c17 --help

1. **プリプロセッサマクロの確認**

* echo | gcc -dM -E -

### 問題1-3: 基本的なプログラム作成

solutions/ex1\_3\_hello\_name.c というファイルを作成し、以下の機能を実装してください。

**要求仕様**

* ユーザーに名前の入力を求める
* 入力された名前を使って挨拶メッセージを表示する
* C90準拠で記述する

**実行例:**

あなたの名前を入力してください: 田中  
こんにちは、田中さん！

**追加の出力例:**

あなたの名前を入力してください: Alice  
こんにちは、Aliceさん！

## 応用問題

### 問題1-4: 規格比較レポート

C90とC99の主な違いについて調べ、以下の点についてまとめてください。

1. **新しく追加されたデータ型**
2. **新しく追加された機能**
3. **プログラムの書き方で変わった点**
4. **組込み開発でC90が使われ続ける理由**

回答は exercises/standards\_comparison.md に記述してください。

### 問題1-5: コンパイル手順の詳細調査

以下のコンパイラオプションを使用して、コンパイル過程を段階的に確認してください。

1. **プリプロセッサ出力の確認**

* gcc -E examples/environment\_check.c > preprocessed.i

1. **アセンブリコード生成**

* gcc -S examples/environment\_check.c

1. **オブジェクトファイル生成**

* gcc -c examples/environment\_check.c

1. **実行ファイル生成**

* gcc environment\_check.o -o environment\_check

各段階で生成されるファイルの内容を確認し、exercises/compilation\_report.md にまとめてください。

### 問題1-6: 環境固有情報の調査

以下の情報を調べて、solutions/ex1\_6\_system\_info.c プログラムを作成してください。

**表示する情報**

* 使用しているOS
* コンパイラの種類とバージョン
* CPUアーキテクチャ（32bit/64bit）
* エンディアン（ビッグエンディアン/リトルエンディアン）
* 各データ型のサイズ

**期待される出力例:**

=== システム情報 ===  
OS: Linux  
コンパイラ: GCC 11.4.0  
アーキテクチャ: 64bit  
エンディアン: リトルエンディアン  
  
=== データ型のサイズ ===  
char: 1 bytes  
short: 2 bytes  
int: 4 bytes  
long: 8 bytes  
long long: 8 bytes  
float: 4 bytes  
double: 8 bytes  
ポインタ: 8 bytes

## チャレンジ問題

### 問題1-7: マルチ規格対応プログラム

以下の要件を満たすプログラムを作成してください。

**要求仕様**

* C90、C99、C11で異なる動作をするプログラム
* プリプロセッサマクロを使って規格を判定
* 各規格で追加された機能を実際に使用
* 規格による違いを分かりやすく表示

**実装例**

* C90: 基本的な機能のみ
* C99: 可変長配列、inline関数を使用
* C11: \_Generic、\_Static\_assertを使用

**期待される出力例:**

=== C規格判定プログラム ===  
  
検出されたC規格: C11  
  
使用可能な機能:  
✓ 基本的なC言語機能  
✓ C99の機能（可変長配列、inline関数）  
✓ C11の機能（\_Generic、\_Static\_assert）  
  
機能デモ:  
配列サイズ 5 の可変長配列を作成しました  
\_Generic による型判定: 整数型です  
\_Static\_assert によるコンパイル時チェック: OK

### 問題1-8: 自動環境診断ツール

開発環境を自動診断するツールを作成してください。

**診断項目**

* コンパイラの種類と詳細バージョン
* 対応している全てのC言語規格
* システム情報（OS、アーキテクチャ、エンディアン）
* 標準ライブラリのバージョン
* 警告レベルごとのコンパイル結果

**出力形式**

* HTMLまたはMarkdown形式のレポート生成
* 問題がある場合は推奨対策を表示

**期待される出力例（コンソール）:**

環境診断ツール v1.0  
===================  
  
診断を開始します...  
  
[✓] コンパイラチェック  
 GCC 11.4.0 が検出されました  
   
[✓] C規格サポート  
 C90: 対応  
 C99: 対応  
 C11: 対応  
 C17: 対応  
 C23: 部分対応  
   
[✓] システム情報  
 OS: Linux 5.15.0  
 アーキテクチャ: x86\_64  
 エンディアン: Little Endian  
   
[!] 警告レベルテスト  
 -Wall: 警告なし  
 -Wextra: 警告なし  
 -pedantic: 1個の警告  
   
診断完了  
レポートを report.html に保存しました

## 提出形式

各問題の回答は以下のファイルに記述してください。

* **問題1-3**: solutions/ex1\_3\_hello\_name.c
* **問題1-4**: exercises/standards\_comparison.md
* **問題1-5**: exercises/compilation\_report.md
* **問題1-6**: solutions/ex1\_6\_system\_info.c
* **問題1-7**: solutions/ex1\_7\_multi\_standard.c
* **問題1-8**: solutions/ex1\_8\_env\_diagnostic.c

## 参考資料

* [GCC公式ドキュメント](https://gcc.gnu.org/documentation.html)
* [C言語規格書](https://www.iso.org/standard/74528.html)
* [GNU Make マニュアル](https://www.gnu.org/software/make/manual/)

## 次の段階

演習課題が完了したら、[基本文法・Hello World](../../basics-syntax/README.md) に進んでください。

# 第2章: 基本文法・Hello World - 演習問題

## 基礎問題

### 問題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

**期待される出力例:**

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

**ファイル名:** ex2\_1\_hello\_intro.c

### 問題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

**期待される出力例:**

=== 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  
================

**ファイル名:** ex2\_2\_personal\_data.c

### 問題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

**期待される出力例:**

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！

**ファイル名:** ex2\_3\_age\_calculator.c

## 応用問題

### 問題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

**期待される出力例:**

=== 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: 3.141590  
 小数点以下2桁: 3.14  
 指数表記: 3.141590e+00  
 フィールド幅10.2: | 3.14|  
  
文字コード 65:  
 文字として: A  
 16進数: 41  
 10進数: 65  
====================

**ファイル名:** ex2\_4\_format\_demo.c

### 問題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

**期待される出力例:**

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: 10  
第2の数値を入力してください: 3  
  
計算結果:  
 10 + 3 = 13  
 10 - 3 = 7  
 10 \* 3 = 30  
 10 / 3 = 3 (整数除算)  
 10 / 3 = 3.33 (実数除算)

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

## チャレンジ問題

### 問題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

**期待される出力例:**

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

**ファイル名:** ex2\_6\_ascii\_explorer.c

### 問題2-7: 文字列の基本操作

文字配列を使って基本的な文字列操作を学ぶプログラムを作成してください。

**機能:**

1. ユーザーから短い文字列（単語）を入力してもらう
2. 文字列の各文字を1文字ずつ表示
3. 文字列を逆順に表示
4. 各文字のASCIIコードを表示

**出力例:**

単語を入力してください: Hello  
  
入力された文字列: Hello  
  
1文字ずつの表示:  
H - e - l - l - o  
  
逆順表示: olleH  
  
各文字のASCIIコード:  
H: 72  
e: 101  
l: 108  
l: 108  
o: 111

**要求事項**

* 文字配列を使用すること
* ループを使って文字を1つずつ処理すること
* 文字列の長さを数える処理を含めること

**期待される出力例:**

単語を入力してください: Hello  
  
入力された文字列: Hello  
  
1文字ずつの表示:  
H - e - l - l - o  
  
逆順表示: olleH  
  
各文字のASCIIコード:  
H: 72  
e: 101  
l: 108  
l: 108  
o: 111

**ファイル名:** ex2\_7\_string\_basics.c

### 問題2-8: プログラマー情報カード

プログラマーの情報を整形して表示するプログラムを作成してください。

**機能:**

1. 複数の情報を入力（名前、年齢、経験年数、好きな言語）
2. 情報をカード形式で綺麗に表示
3. 経験レベルを判定（初心者/中級者/上級者）

**出力例:**

=== プログラマー情報入力 ===  
名前: 山田太郎  
年齢: 25  
プログラミング経験年数: 3  
好きなプログラミング言語: C  
  
+------------------------------------+  
| プログラマー情報カード |  
+------------------------------------+  
| 名前: 山田太郎 |  
| 年齢: 25歳 |  
| 経験: 3年 |  
| レベル: 中級者 |  
| 好きな言語: C |  
+------------------------------------+  
  
プログラミングの学習、頑張ってください！

**経験レベルの判定基準**

* 0-1年: 初心者
* 2-4年: 中級者
* 5年以上: 上級者

**要求事項**

* 適切な変数型を使用すること
* 条件分岐で経験レベルを判定すること
* 整形された出力を作成すること

**期待される出力例:**

=== プログラマー情報入力 ===  
名前: 山田太郎  
年齢: 25  
プログラミング経験年数: 3  
好きなプログラミング言語: C  
  
+------------------------------------+  
| プログラマー情報カード |  
+------------------------------------+  
| 名前: 山田太郎 |  
| 年齢: 25歳 |  
| 経験: 3年 |  
| レベル: 中級者 |  
| 好きな言語: C |  
+------------------------------------+  
  
プログラミングの学習、頑張ってください！

**ファイル名:** ex2\_8\_programmer\_card.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  
├── ex2\_7\_string\_basics.c  
└── ex2\_8\_programmer\_card.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！

# 第3章: データ型と変数 - 演習問題

## 基礎問題

### 演習3-1. 変数宣言と初期化

さまざまなデータ型の変数を宣言し、初期化して値を表示するプログラムを作成してください。

**要件**

* 各基本データ型（char, short, int, long, float, double）の変数を宣言
* 適切な値で初期化
* printf関数で型に応じた書式指定子を使用して出力
* 符号付き・符号なしの両方を含める

**期待される出力例:**

char型: 文字 = 'A', 値 = 65  
unsigned char型: 値 = 255  
short型: 値 = -1000  
unsigned short型: 値 = 65535  
int型: 値 = -123456  
unsigned int型: 値 = 4294967295  
float型: 値 = 3.14159  
double型: 値 = 3.141592653589793

### 演習3-2. 四則演算計算機

2つの数値を変数に格納し、四則演算（加算、減算、乗算、除算）の結果を表示するプログラムを作成してください。

**要件**

* 整数型と浮動小数点型の両方で実装
* 除算では整数除算と実数除算の違いを表示
* 各演算結果を見やすく表示

**期待される出力例:**

=== 整数演算 ===  
a = 15, b = 4  
加算: 15 + 4 = 19  
減算: 15 - 4 = 11  
乗算: 15 \* 4 = 60  
除算: 15 / 4 = 3 (整数除算)  
  
=== 実数演算 ===  
x = 15.0, y = 4.0  
加算: 15.00 + 4.00 = 19.00  
減算: 15.00 - 4.00 = 11.00  
乗算: 15.00 \* 4.00 = 60.00  
除算: 15.00 / 4.00 = 3.75

### 演習3-3. データ型サイズの確認

sizeof演算子を使って、各データ型のサイズを表示するプログラムを作成してください。

**要件**

* 基本データ型すべてのサイズを表示
* 配列のサイズも確認
* ポインタのサイズも確認
* サイズをバイト単位で表示

**期待される出力例:**

=== データ型のサイズ ===  
char: 1 バイト  
short: 2 バイト  
int: 4 バイト  
long: 8 バイト  
float: 4 バイト  
double: 8 バイト  
  
=== 配列のサイズ ===  
int array[10]: 40 バイト  
要素数: 10  
  
=== ポインタのサイズ ===  
char\*: 8 バイト  
int\*: 8 バイト  
double\*: 8 バイト

## 応用問題

### 演習3-4. 型変換の理解

整数除算と実数除算の違い、および暗黙的・明示的型変換を確認するプログラムを作成してください。

**要件**

* 同じ数値で整数除算と実数除算を実行し、結果を比較
* 暗黙的型変換が発生する例を実装
* 明示的型変換（キャスト）を使用した例を実装
* 精度の損失が発生する例を示す

**期待される出力例:**

=== 整数除算と実数除算 ===  
int a = 10, b = 3  
整数除算: 10 / 3 = 3  
実数除算（キャスト使用）: 10 / 3 = 3.333333  
  
=== 暗黙的型変換 ===  
int i = 10  
double d = 3.14  
i + d = 13.140000 (intがdoubleに変換)  
  
=== 明示的型変換（キャスト） ===  
double x = 3.75  
(int)x = 3 (小数部分が切り捨て)  
  
=== 精度の損失 ===  
float f = 1234567890.123456  
表示: 1234567936.000000 (精度が失われる)

### 演習3-5. スコープの実験

グローバル変数、ローカル変数、静的変数を使い分けるプログラムを作成してください。

**要件**

* 同じ名前の変数を異なるスコープで宣言
* スコープの隠蔽（シャドウイング）を実演
* 静的変数を使ったカウンター関数を実装
* 各変数の値の変化を追跡して表示

**期待される出力例:**

=== スコープの実験 ===  
グローバル変数 x = 100  
  
main関数内:  
ローカル変数 x = 10 (グローバル変数を隠蔽)  
  
ブロック内:  
ブロック内の x = 5 (main関数の変数を隠蔽)  
ブロックを出た後の x = 10  
  
=== 静的変数カウンター ===  
counter()の呼び出し 1回目: 1  
counter()の呼び出し 2回目: 2  
counter()の呼び出し 3回目: 3  
  
グローバル変数 x = 100 (変更されていない)

### 演習3-6. 定数の活用

constと#defineを使って定数を定義し、円の面積と円周を計算するプログラムを作成してください。

**要件**

* 円周率をconstと#defineの両方で定義
* 半径を入力として受け取る
* 面積と円周を計算して表示
* 定数を変更しようとした場合のエラーを確認（コメントで説明）

**期待される出力例:**

円の計算プログラム  
=================  
半径を入力してください: 5.0  
  
計算結果:  
半径: 5.00  
円周: 31.42 (#define版の円周率使用)  
面積: 78.54 (const版の円周率使用)  
  
両方の円周率の値:  
#define PI = 3.141593  
const pi = 3.141593

## チャレンジ問題

### 演習3-7. 温度変換プログラム

摂氏・華氏・ケルビンの温度単位を相互変換するプログラムを作成してください。

**要件**

* 変換式で使用する定数を適切に定義
* 浮動小数点演算の精度に注意
* ユーザーフレンドリーな入出力
* 変換公式。
  + 華氏 = 摂氏 × 9/5 + 32
  + ケルビン = 摂氏 + 273.15

**期待される出力例:**

=== 温度変換プログラム ===  
温度を入力してください（摂氏）: 25.0  
  
変換結果:  
摂氏: 25.00 °C  
華氏: 77.00 °F  
ケルビン: 298.15 K  
  
別の例（氷点）:  
摂氏: 0.00 °C  
華氏: 32.00 °F  
ケルビン: 273.15 K  
  
別の例（沸点）:  
摂氏: 100.00 °C  
華氏: 212.00 °F  
ケルビン: 373.15 K

## 提出方法

1. 各問題に対して別々のCファイルを作成
2. ファイル名は ex3\_1\_variables.c, ex3\_2\_calculator.c のように命名
3. 各プログラムの冒頭にコメントで問題番号と簡単な説明を記載
4. C90規格でコンパイルできることを確認

## コンパイル例

gcc -std=c90 -Wall -Wextra -pedantic ex\_variables.c -o ex\_variables  
./ex\_variables

## ヒント

* printf関数の書式指定子を正しく使用する
  + %d: int
  + %u: unsigned int
  + %ld: long
  + %f: float/double
  + %c: char
* 初期化されていない変数の使用に注意
* 型の範囲を超えないよう注意
* sizeof演算子の戻り値はsize\_t型（%luで出力）

# 第4章: 演算子 - 演習問題

## 基礎問題

### 問題4-1: 四則演算計算機

2つの整数を入力として受け取り、すべての算術演算（加算、減算、乗算、除算、剰余）の結果を表示するプログラムを作成してください。

**要件**

* ユーザーから2つの整数を入力
* 各演算の結果を見やすく表示
* ゼロ除算のチェックを含める

**期待される出力例:**

2つの整数を入力してください: 10 3  
10 + 3 = 13  
10 - 3 = 7  
10 \* 3 = 30  
10 / 3 = 3  
10 % 3 = 1

**ファイル名:** ex4\_1\_calculator.c

### 問題4-2: 比較と論理演算

2つの整数を入力として受け取り、関係演算子と論理演算子を使って最大値と最小値を求めるプログラムを作成してください。

**要件**

* 2つの整数を入力
* if文を使わず、条件演算子（三項演算子）のみで実装
* 最大値と最小値を表示

**期待される出力例:**

2つの整数を入力してください: 15 8  
最大値: 15  
最小値: 8  
  
別の例:  
2つの整数を入力してください: -5 3  
最大値: 3  
最小値: -5

**ファイル名:** ex4\_2\_comparison.c

### 問題4-3: インクリメント・デクリメント

前置と後置のインクリメント・デクリメント演算子の違いを確認するプログラムを作成してください。

**要件**

* 変数に対して前置・後置の両方を使用
* 各操作後の値を表示
* 式の中での動作も確認

**期待される出力例:**

=== インクリメント・デクリメント演算子の動作確認 ===  
初期値: x = 5  
  
後置インクリメント (x++): 5  
操作後の x: 6  
  
前置インクリメント (++x): 7  
操作後の x: 7  
  
初期値: y = 10  
式の中での動作:  
a = y++ + 2 = 12 (y = 11)  
b = ++y + 2 = 14 (y = 12)  
  
デクリメントも同様:  
初期値: z = 8  
c = z-- = 8 (z = 7)  
d = --z = 6 (z = 6)

**ファイル名:** ex4\_3\_increment.c

## 応用問題

### 問題4-4: ビット操作

整数を入力として受け取り、その数値の各ビットを表示し、特定のビット操作を実行するプログラムを作成してください。

**要件**

* 2進数表示
* 指定したビット位置のON/OFF確認
* 特定ビットの設定・クリア・トグル機能

**期待される出力例:**

整数を入力してください: 10  
10 の2進数表示: 00001010  
  
ビット位置を入力してください（0-7）: 2  
ビット位置 2 は OFF です  
  
操作を選択してください:  
1. ビットを設定（ON）  
2. ビットをクリア（OFF）  
3. ビットをトグル（反転）  
選択: 1  
  
操作後: 14 (00001110)  
ビット位置 2 を ON にしました

**ファイル名:** ex4\_4\_bitwise.c

### 問題4-5: 条件演算子の活用

3つの数値を入力として受け取り、条件演算子（三項演算子）のみを使って昇順に並び替えるプログラムを作成してください。

**要件**

* if文、switch文を使用しない
* 条件演算子のネストを活用
* 並び替えた結果を表示

**期待される出力例:**

3つの整数を入力してください: 9 3 6  
昇順に並び替え: 3 6 9  
  
別の例:  
3つの整数を入力してください: -5 10 0  
昇順に並び替え: -5 0 10

**ファイル名:** ex4\_5\_ternary\_sort.c

### 問題4-6: 演算子優先順位

複雑な式を含むプログラムを作成し、演算子の優先順位による計算結果の違いを確認してください。

**要件**

* 同じ数値で異なる式を評価
* 括弧の有無による結果の違いを表示
* 少なくとも3種類以上の演算子を使用

**期待される出力例:**

=== 演算子優先順位の確認 ===  
a = 2, b = 3, c = 4  
  
式1: a + b \* c = 14  
説明: 乗算が先 → 2 + (3 \* 4) = 2 + 12 = 14  
  
式2: (a + b) \* c = 20  
説明: 括弧内が先 → (2 + 3) \* 4 = 5 \* 4 = 20  
  
式3: a << b + c = 256  
説明: 加算が先 → 2 << (3 + 4) = 2 << 7 = 256  
  
式4: a & b | c = 6  
説明: &が先、次に| → (2 & 3) | 4 = 2 | 4 = 6

**ファイル名:** ex4\_6\_precedence.c

## チャレンジ問題

### 問題4-7: 複合代入演算子の応用

複合代入演算子（+=, -=, \*=, /=, %=, &=, |=, ^=, <<=, >>=）をすべて使用して、数値の変換処理を行うプログラムを作成してください。

**要件**

* 初期値から各種複合代入演算子を使って変換
* 各ステップでの値の変化を表示
* ビット演算の複合代入も含める

**期待される処理:**

1. 初期値を入力
2. 各種演算を順番に適用
3. 最終結果と各ステップを表示

**ファイル名:** ex4\_7\_compound\_operators.c

### 問題4-8: ビット演算による最適化

ビット演算を使って、以下の処理を高速に実行するプログラムを作成してください。

**実装する機能:**

1. 2の累乗かどうかの判定
2. 2の累乗による乗算・除算
3. 偶数・奇数の判定
4. 数値の符号反転（2の補数）
5. 最下位の1ビットを抽出

**要件**

* 通常の演算子を使わずビット演算のみで実装
* 各機能の動作原理をコメントで説明
* テストケースを含める

**ファイル名:** ex4\_8\_bit\_optimization.c

### 問題4-9: 演算子組み合わせパズル

与えられた数値と演算子を使って、目標の値を作るプログラムを作成してください。

**仕様:**

1. 4つの1桁の数字を入力
2. 目標値を入力
3. +, -, \*, / の演算子を使用
4. 括弧の使用も可能
5. すべての数字を1回ずつ使用
6. 解が存在する場合は式を表示

**例:**

数字を4つ入力: 2 3 4 5  
目標値: 24  
解: (2 + 3) \* 4 + 5 - 5 = 24

**ファイル名:** ex4\_9\_operator\_puzzle.c

## 提出方法

1. 各問題に対して別々のCファイルを作成
2. ファイル名は指定された名前を使用
3. 各プログラムの冒頭にコメントで問題番号と簡単な説明を記載
4. C90規格でコンパイルできることを確認

## コンパイル例

gcc -std=c90 -Wall -Wextra -pedantic ex4\_1\_calculator.c -o ex4\_1\_calculator  
./ex4\_1\_calculator

## 評価ポイント

* **正確性**: 要求された機能が正しく動作するか
* **エラー処理**: ゼロ除算などの例外的な状況への対処
* **コード品質**: 可読性、適切なコメント、変数名
* **効率性**: 特にビット演算問題での最適化

## ヒント

* 演算子の優先順位に注意
* ゼロ除算やオーバーフローに対する考慮
* 可読性を意識したコードを心がける
* 適切なコメントを追加する
* ビット演算では2進数表現を意識する

# 第5章: 制御構造（条件分岐） - 演習問題

## 演習の目的

* if文、switch文の理解を深める
* 条件演算子の適切な使い方を習得する
* 複雑な条件分岐の設計力を養う

## 基礎問題

### 問題5-1: 年齢による料金計算

映画館の料金システムを実装してください。

**要件**

* 3歳未満: 無料
* 3歳以上12歳以下: 子供料金（800円）
* 13歳以上18歳以下: 学生料金（1200円）
* 19歳以上65歳以下: 大人料金（1800円）
* 66歳以上: シニア料金（1200円）

**実装のヒント**

* if-else if文を使用
* 境界値に注意

**期待される出力例:**

映画館料金計算システム  
年齢を入力してください: 25  
お客様の料金は 1800円 です（大人料金）  
  
別の例:  
年齢を入力してください: 5  
お客様の料金は 800円 です（子供料金）  
  
年齢を入力してください: 70  
お客様の料金は 1200円 です（シニア料金）

**ファイル名:** ex5\_1\_movie\_price.c

### 問題5-2: 電卓プログラム

四則演算ができる簡単な電卓を作成してください。

**要件**

* ユーザーから2つの数値と演算子（+, -, \*, /）を入力
* switch文を使って演算を実行
* ゼロ除算のエラー処理を含める

**実装のヒント**

* 演算子は char 型で受け取る
* 除算の場合は分母が0でないかチェック

**期待される出力例:**

簡単な電卓プログラム  
第1の数値を入力: 15  
演算子を入力 (+, -, \*, /): \*  
第2の数値を入力: 3  
計算結果: 15.00 \* 3.00 = 45.00  
  
別の例（ゼロ除算）:  
第1の数値を入力: 10  
演算子を入力 (+, -, \*, /): /  
第2の数値を入力: 0  
エラー: ゼロで除算することはできません

**ファイル名:** ex5\_2\_calculator.c

### 問題5-3: BMI判定プログラム

身長と体重からBMIを計算し、判定するプログラムを作成してください。

**要件**

* BMI = 体重(kg) / (身長(m) × 身長(m))
* BMI判定基準:
  + 18.5未満: 低体重
  + 18.5以上25未満: 標準体重
  + 25以上30未満: 肥満度1
  + 30以上: 肥満度2

**実装のヒント**

* 身長はセンチメートルで入力してメートルに変換
* 浮動小数点数を使用

**期待される出力例:**

BMI判定プログラム  
身長(cm)を入力してください: 170  
体重(kg)を入力してください: 65  
  
BMI値: 22.49  
判定: 標準体重です  
  
別の例:  
身長(cm)を入力してください: 165  
体重(kg)を入力してください: 80  
  
BMI値: 29.38  
判定: 肥満度1です

**ファイル名:** ex5\_3\_bmi\_check.c

## 応用問題

### 問題5-4: うるう年判定

西暦年を入力してうるう年かどうか判定するプログラムを作成してください。

**うるう年の条件:**

1. 4で割り切れる年はうるう年
2. ただし、100で割り切れる年は平年
3. ただし、400で割り切れる年はうるう年

**実装のヒント**

* 複数の条件を論理演算子で組み合わせる
* 条件の順序に注意

**期待される出力例:**

うるう年判定プログラム  
西暦年を入力してください: 2024  
2024年はうるう年です  
  
別の例:  
西暦年を入力してください: 1900  
1900年は平年です  
  
西暦年を入力してください: 2000  
2000年はうるう年です

**ファイル名:** ex5\_4\_leap\_year.c

### 問題5-5: 成績評価システム

複数の科目の点数から総合評価を出すプログラムを作成してください。

**要件**

* 3科目（国語、数学、英語）の点数を入力
* 平均点を計算
* 平均点による評価（A〜F）
* 全科目60点以上の場合のみ「合格」、それ以外は「不合格」
* 条件演算子を使って簡潔に表示

**評価基準**

* 90点以上: A
* 80点以上90点未満: B
* 70点以上80点未満: C
* 60点以上70点未満: D
* 50点以上60点未満: E
* 50点未満: F

**実装のヒント**

* 平均点の計算は整数演算に注意
* 複数の条件を組み合わせる

**期待される出力例:**

成績評価システム  
国語の点数: 85  
数学の点数: 92  
英語の点数: 78  
  
平均点: 85.00  
評価: B  
判定: 合格  
  
別の例（不合格）:  
国語の点数: 75  
数学の点数: 55  
英語の点数: 80  
  
平均点: 70.00  
評価: C  
判定: 不合格（数学が60点未満）

**ファイル名:** ex5\_5\_grade\_system.c

### 問題5-6: 曜日判定プログラム

年月日を入力して、その日の曜日を判定するプログラムを作成してください。

**要件**

* ツェラーの公式を使用して曜日を計算
* 入力値の妥当性チェック（月は1-12、日は1-31など）
* 曜日を日本語で表示

**ツェラーの公式（簡易版）**

* 1月、2月は前年の13月、14月として計算

**期待される出力例:**

曜日判定プログラム  
年を入力: 2024  
月を入力: 3  
日を入力: 15  
  
2024年3月15日は金曜日です  
  
別の例:  
年を入力: 2023  
月を入力: 12  
日を入力: 25  
  
2023年12月25日は月曜日です

**ファイル名:** ex5\_6\_weekday.c

## チャレンジ問題

### 問題5-7: じゃんけんゲーム

コンピュータとじゃんけんをするプログラムを作成してください。

**要件**

* ユーザーの手を数値で入力（1:グー、2:チョキ、3:パー）
* コンピュータの手はランダムに決定
* 勝敗を判定して表示
* 不正な入力のチェック

**追加要件**

* 3回勝負で最終的な勝者を決定
* 各回の結果を記録して最後に表示
* 連勝ボーナスポイントシステム

**期待される出力例:**

=== じゃんけんゲーム（3回勝負） ===  
  
第1回戦  
あなたの手を選んでください（1:グー、2:チョキ、3:パー）: 1  
あなた: グー  
コンピュータ: チョキ  
結果: あなたの勝ち！  
  
第2回戦  
あなたの手を選んでください（1:グー、2:チョキ、3:パー）: 2  
あなた: チョキ  
コンピュータ: チョキ  
結果: あいこ  
  
第3回戦  
あなたの手を選んでください（1:グー、2:チョキ、3:パー）: 3  
あなた: パー  
コンピュータ: グー  
結果: あなたの勝ち！（連勝ボーナス +1）  
  
=== 最終結果 ===  
あなた: 3ポイント（2勝0敗1分）  
コンピュータ: 0ポイント  
優勝: あなた！

**ファイル名:** ex5\_7\_janken\_game.c

### 問題5-8: 交通信号シミュレーター

信号機の動作をシミュレートするプログラムを作成してください。

**要件**

* 現在の信号の色と経過時間から次の動作を決定
* 歩行者用信号も含める
* 緊急車両通過モード（すべて赤信号）
* 時間帯による点滅信号への切り替え

**信号のパターン**

* 青（30秒）→ 黄（3秒）→ 赤（30秒）
* 歩行者: 青（20秒）→ 点滅（10秒）→ 赤（33秒）

**期待される出力例:**

交通信号シミュレーター  
現在の信号状態を入力（1:青、2:黄、3:赤）: 1  
経過時間（秒）: 28  
現在時刻（0-23時）: 14  
  
=== 信号状態 ===  
車両用信号: 青（残り2秒）  
歩行者信号: 赤  
次の状態: まもなく黄信号に変わります  
  
別の例（深夜）:  
現在時刻（0-23時）: 2  
深夜モード: 黄色点滅信号  
  
別の例（緊急車両）:  
緊急車両通過モード（1:有効、0:無効）: 1  
すべての信号: 赤（緊急車両通過中）

**ファイル名:** ex5\_8\_traffic\_light.c

### 問題5-9: 複雑な料金計算システム

タクシー料金計算システムを実装してください。

**要件**

* 初乗り料金: 1.052kmまで420円
* 加算料金: 以降237mごとに80円
* 時間距離併用: 時速10km以下では90秒ごとに80円
* 深夜割増（22:00-5:00）: 2割増
* 予約料金、高速料金の加算オプション

**入力項目**

* 走行距離、走行時間、時刻、オプション選択

**期待される出力例:**

タクシー料金計算システム  
走行距離(km): 5.5  
走行時間(分): 25  
現在時刻(0-23): 23  
予約利用(1:あり、0:なし): 0  
高速道路利用(1:あり、0:なし): 0  
  
=== 料金明細 ===  
初乗り料金: 420円  
加算料金: 760円（4.448km分）  
深夜割増（20%）: 236円  
-----------------  
合計料金: 1,416円  
  
別の例（渋滞時）:  
走行距離(km): 2.0  
走行時間(分): 30  
現在時刻(0-23): 18  
  
=== 料金明細 ===  
初乗り料金: 420円  
加算料金: 320円（距離加算）  
時間加算料金: 640円（低速走行）  
-----------------  
合計料金: 1,380円

**ファイル名:** ex5\_9\_taxi\_fare.c

## 提出方法

1. 各問題に対して個別のCファイルを作成
2. ファイル名は指定された名前を使用
3. コメントで問題番号と簡単な説明を記載
4. コンパイル・実行確認を必ず行う

## 評価基準

* **正確性**: 要求仕様を満たしているか
* **可読性**: 適切なインデント、変数名、コメント
* **効率性**: 不要な条件分岐がないか
* **エラー処理**: 不正な入力への対処

## 学習のポイント

* 条件の順序と網羅性
* switch文とif文の使い分け
* 条件演算子による簡潔な記述
* 複雑な条件の論理的な整理

# 第6章: 制御構造（ループ） - 演習問題

## 演習の目的

* for文、while文、do-while文の使い分けを理解する
* break文とcontinue文を適切に使えるようになる
* ネストしたループを活用した問題解決能力を養う

## 基礎問題

### 問題6-1: フィボナッチ数列

最初のN個のフィボナッチ数を表示するプログラムを作成してください。

**要件**

* ユーザーから表示する個数Nを入力
* フィボナッチ数列: 0, 1, 1, 2, 3, 5, 8, 13, …
* for文を使用して実装

**実装のヒント**

* 最初の2つの数は0と1
* 3番目以降は前の2つの数の和

**期待される出力例:**

フィボナッチ数列表示プログラム  
表示する個数を入力してください: 10  
  
フィボナッチ数列（最初の10個）:  
0 1 1 2 3 5 8 13 21 34  
  
別の例:  
表示する個数を入力してください: 15  
  
フィボナッチ数列（最初の15個）:  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

**ファイル名:** ex6\_1\_fibonacci.c

### 問題6-2: 数当てゲーム

1から100までの数当てゲームを作成してください。

**要件**

* プログラムがランダムに1～100の数を決定
* ユーザーが予想を入力
* 「もっと大きい」「もっと小さい」のヒントを出す
* 10回以内に当てられなければゲームオーバー
* do-while文を使用

**実装のヒント**

* rand() % 100 + 1 でランダムな数を生成
* 試行回数をカウント

**期待される出力例:**

=== 数当てゲーム ===  
1から100までの数を当ててください！  
10回以内に当てられればクリアです。  
  
あなたの予想: 50  
もっと小さい数です  
  
あなたの予想: 25  
もっと大きい数です  
  
あなたの予想: 37  
もっと大きい数です  
  
あなたの予想: 43  
もっと小さい数です  
  
あなたの予想: 40  
正解です！  
4回で当てました。おめでとう！  
  
別の例（失敗）:  
（10回予想しても当たらない場合）  
残念！答えは 72 でした。

**ファイル名:** ex6\_2\_number\_game.c

### 問題6-3: 掛け算表

9×9の掛け算表を表示するプログラムを作成してください。

**要件**

* ネストしたfor文を使用
* 見やすい表形式で出力
* 列と行のヘッダーを含める

**出力例:**

1 2 3 4 5 6 7 8 9  
1 1 2 3 4 5 6 7 8 9  
2 2 4 6 8 10 12 14 16 18  
3 3 6 9 12 15 18 21 24 27  
...

**実装のヒント**

* printf()の書式指定子で桁揃えを行う
* %3d などを使用して桁数を統一

**ファイル名:** ex6\_3\_multiplication\_table.c

## 応用問題

### 問題6-4: 素数リスト

指定された範囲内のすべての素数を表示するプログラムを作成してください。

**要件**

* ユーザーから範囲（開始値と終了値）を入力
* その範囲内の素数をすべて表示
* 素数の個数も表示
* ネストしたループとbreak文を活用

**実装のヒント**

* 2からsqrt(n)まで割り切れるか確認
* 効率化のため、偶数は2以外スキップ

**期待される出力例:**

素数リスト表示プログラム  
開始値を入力してください: 10  
終了値を入力してください: 50  
  
10から50までの素数:  
11 13 17 19 23 29 31 37 41 43 47  
  
素数の個数: 11個  
  
別の例:  
開始値を入力してください: 1  
終了値を入力してください: 30  
  
1から30までの素数:  
2 3 5 7 11 13 17 19 23 29  
  
素数の個数: 10個

**ファイル名:** ex6\_4\_prime\_list.c

### 問題6-5: 図形の描画

アスタリスク(\*)を使って様々な図形を描画するプログラムを作成してください。

**要件:** 以下の4つの図形を描画する機能を実装: 1. 正方形 2. 直角三角形 3. 逆直角三角形 4. ダイヤモンド

**例（サイズ5の場合）:**

正方形:

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

直角三角形:

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

逆直角三角形:

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

ダイヤモンド:

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

**実装のヒント**

* ネストしたループを使用
* 空白文字とアスタリスクの配置に注意

**ファイル名:** ex6\_5\_draw\_shapes.c

### 問題6-6: 完全数の探索

完全数を探すプログラムを作成してください。

**完全数とは:** その数自身を除く約数の和が、その数自身と等しくなる自然数 例: 6 = 1 + 2 + 3, 28 = 1 + 2 + 4 + 7 + 14

**要件**

* ユーザーから上限値を入力
* 1からその値までの完全数をすべて表示
* 各完全数について、約数も表示

**期待される出力例:**

完全数探索プログラム  
上限値を入力してください: 1000  
  
1から1000までの完全数:  
  
6は完全数です  
約数: 1 + 2 + 3 = 6  
  
28は完全数です  
約数: 1 + 2 + 4 + 7 + 14 = 28  
  
496は完全数です  
約数: 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248 = 496  
  
見つかった完全数の個数: 3個

**ファイル名:** ex6\_6\_perfect\_numbers.c

## チャレンジ問題

### 問題6-7: パスカルの三角形

パスカルの三角形を表示するプログラムを作成してください。

**要件**

* ユーザーから行数を入力（最大15行）
* 適切な配置で三角形を表示
* 各数値は前の行の2つの数の和

**出力例（5行の場合）:**

1  
 1 1  
 1 2 1  
 1 3 3 1  
1 4 6 4 1

**実装のヒント**

* 2次元配列を使用
* 適切な空白で中央揃え

**ファイル名:** ex6\_7\_pascal\_triangle.c

### 問題6-8: コンソールアニメーション

簡単なコンソールアニメーションを作成してください。

**要件**

* 横に移動するアスタリスクのアニメーション
* 画面幅を考慮して左右に往復
* ESCキーまたはCtrl+Cで終了
* system(“clear”)またはsystem(“cls”)を使用

**動作例:**

\* （右に移動）  
 \*  
 \*  
 \*  
 \* （端に到達したら折り返し）  
 \*  
 \*  
 \*  
\* （左端に戻る）

**期待される出力例:**

コンソールアニメーション  
Ctrl+Cで終了します...  
  
（画面がクリアされて以下のアニメーションが繰り返される）  
  
フレーム1:  
\*  
  
フレーム2:  
 \*  
  
フレーム3:  
 \*  
  
... (続く) ...  
  
フレーム40:  
 \*  
  
（端に到達後、逆方向に移動）  
  
フレーム41:  
 \*  
  
... (左端まで戻って繰り返し) ...

**ファイル名:** ex6\_8\_console\_animation.c

### 問題6-9: 数字ピラミッド

数字を使ったピラミッドパターンを生成するプログラムを作成してください。

**要件**

* ユーザーから高さを入力（1-9）
* 複数のパターンを選択可能
* continue文を活用した効率的な実装

**パターン例（高さ5）:**

パターン1:

1  
 2 2  
 3 3 3  
 4 4 4 4  
5 5 5 5 5

パターン2:

1  
 1 2 1  
 1 2 3 2 1  
 1 2 3 4 3 2 1  
1 2 3 4 5 4 3 2 1

パターン3:

1  
 2 3 2  
 3 4 5 4 3  
 4 5 6 7 6 5 4  
5 6 7 8 9 8 7 6 5

**ファイル名:** ex6\_9\_number\_pyramid.c

## 提出方法

1. 各問題に対して個別のCファイルを作成
2. ファイル名は指定された名前を使用
3. コメントで問題番号と簡単な説明を記載
4. コンパイル・実行確認を必ず行う

## 評価基準

* **正確性**: 要求仕様を満たしているか
* **効率性**: 適切なループの選択と最適化
* **可読性**: 適切なインデント、変数名、コメント
* **創造性**: チャレンジ問題での工夫

## 学習のポイント

* for、while、do-whileの使い分け
* breakとcontinueの適切な使用
* ネストループの理解と活用
* ループの効率化技術

# 演習問題を解き終えたら

## チェックリスト

基礎編の演習問題をすべて解き終えたら、以下の点を確認してください：

* すべての問題でコンパイルエラーなく実行できた
* 期待される出力が得られた
* コードの意味を説明できる
* 別の方法での実装を考えられる

## 次のステップ

基礎編の演習を完了したら、応用編の学習に進んでください。応用編では、より複雑なデータ構造とアルゴリズムを扱います。

## 困ったときは

* メインテキストの該当章を再度確認
* サンプルコードと比較
* エラーメッセージを注意深く読む
* 段階的にデバッグ出力を追加

プログラミングは試行錯誤の連続です。エラーを恐れず、積極的に挑戦してください。