# 第1章 導入・環境構築

## 対応C規格

* **主要対象:** 全規格共通
* **学習内容:** C言語の基礎知識、開発環境構築、規格解説

## 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：

* C言語の特徴と歴史を理解する
* 開発環境を構築し、基本的なコンパイル手順を覚える
* C言語の規格（C90, C99, C11, C17）の違いを理解する
* 簡単なプログラムをコンパイル・実行できる

## 理論解説

### C言語とは

C言語は1972年にデニス・リッチーがベル研究所で開発したプログラミング言語です。以下の特徴があります：

#### 主な特徴

* **システムプログラミング向け**: OS開発やハードウェア制御に適している
* **高い移植性**: 異なるプラットフォーム間でのコード再利用が容易
* **効率性**: メモリとCPUの使用効率が良い
* **シンプルな文法**: 基本的な機能に絞られた明確な仕様

#### 使用分野

* オペレーティングシステム（Linux、Windows等）
* 組込みシステム（マイコン、IoTデバイス）
* データベースシステム
* ゲームエンジン
* 科学技術計算

### C言語の規格

C言語は時代とともに進化し、複数の規格が策定されています：

| 規格名 | 発表年 | 正式名称 | 主な特徴 |
| --- | --- | --- | --- |
| **C90** | 1990年 | ISO/IEC 9899:1990 | 初の国際標準、ANSI C |
| **C99** | 1999年 | ISO/IEC 9899:1999 | \_Bool型、可変長配列、inline関数 |
| **C11** | 2011年 | ISO/IEC 9899:2011 | 匿名構造体、\_Generic、マルチスレッド |
| **C17** | 2018年 | ISO/IEC 9899:2018 | C11のバグ修正版 |

#### 規格選択の指針

* **学習目的**: C90から始めて段階的に新機能を学習
* **業務開発**: C99以降を推奨（実用的な機能が多い）
* **組込み**: C90またはC99（コンパイラ対応状況による）

### 開発環境

C言語プログラムの開発には以下のツールが必要です：

#### 必須ツール

1. **テキストエディター**: コードを記述するツール
2. **コンパイラ**: ソースコードを実行ファイルに変換
3. **リンカー**: 複数のオブジェクトファイルを結合
4. **デバッガー**: プログラムの動作を詳細に調査

#### 推奨開発環境

**Windows:**

# MinGW-w64のインストール（推奨）  
# または Microsoft Visual Studio Community

**macOS:**

# Xcode Command Line Toolsのインストール  
xcode-select --install

**Linux（Ubuntu/Debian）:**

sudo apt update  
sudo apt install build-essential

### コンパイル手順

C言語プログラムの実行までの流れ：

1. **ソースコード作成**（.cファイル）
2. **プリプロセッサ処理**（#includeの展開等）
3. **コンパイル**（アセンブリコードに変換）
4. **アセンブル**（オブジェクトファイルに変換）
5. **リンク**（実行ファイルの生成）
6. **実行**

#### 基本的なコンパイルコマンド

# 一段階でコンパイル  
gcc program.c -o program  
  
# 段階的なコンパイル  
gcc -c program.c # オブジェクトファイル作成  
gcc program.o -o program # 実行ファイル作成

## サンプルコード

### 環境確認プログラム

プログラムファイル: examples/environment\_check.c

このプログラムは開発環境が正しく設定されているかを確認します。

### 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 environment\_check.c -o env\_check\_c90  
  
# C99準拠でコンパイル  
gcc -std=c99 -Wall -Wextra -pedantic environment\_check.c -o env\_check\_c99  
  
# C11準拠でコンパイル  
gcc -std=c11 -Wall -Wextra -pedantic environment\_check.c -o env\_check\_c11  
  
# C17準拠でコンパイル  
gcc -std=c17 -Wall -Wextra -pedantic environment\_check.c -o env\_check\_c17  
  
# 実行  
./env\_check\_c90

## 演習課題

### 基礎問題

1. **環境構築確認**
   * サンプルプログラムを正常にコンパイル・実行できることを確認してください
   * C90、C99、C11、C17の各規格でコンパイルを試してください
2. **コンパイラ情報の確認**
   * 使用しているコンパイラの名前とバージョンを調べてください
   * 対応している規格を確認してください

### 応用問題

1. **規格比較**
   * C90とC99の主な違いを3つ以上挙げてください
   * 組込み開発でC90が今でも使われる理由を考えてください
2. **コンパイル手順の理解**
   * プリプロセッサ、コンパイラ、リンカーの役割を説明してください
   * -E, -S, -c オプションの効果を確認してください

## 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  
  
# C90準拠でコンパイル  
make STANDARD=c90  
  
# C99準拠でコンパイル   
make STANDARD=c99  
  
# C11準拠でコンパイル  
make STANDARD=c11  
  
# C17準拠でコンパイル  
make STANDARD=c17  
  
# クリーンアップ  
make clean

## 規格による違い

この章では基本的な環境確認のため、規格による大きな違いはありません。ただし、コンパイラの警告やエラーメッセージに違いが出る場合があります。

## 次の章へ

環境構築が完了したら、[基本文法・Hello World](../basics-syntax/README.md) に進んでください。

## 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 (ISO/IEC 9899:2018)](https://www.iso.org/standard/74528.html)
* [GNU Make マニュアル](https://www.gnu.org/software/make/manual/)

# サンプルコード

## environment\_check.c

/\*  
 \* environment\_check.c - C言語開発環境確認プログラム  
 \*  
 \* このプログラムは開発環境が正しく設定されているかを確認します。  
 \* 各種C規格でコンパイルして動作を確認してください。  
 \*/  
  
#include <stdio.h>  
#include <limits.h>  
  
/\*  
 \* 環境情報を表示する関数  
 \*/  
void print\_environment\_info(void)  
{  
 printf("=== C言語開発環境確認 ===\n\n");  
  
 /\* コンパイラ情報 \*/  
 printf("コンパイラ情報:\n");  
  
#ifdef \_\_GNUC\_\_  
 printf("- GCC version: %d.%d.%d\n", \_\_GNUC\_\_, \_\_GNUC\_MINOR\_\_, \_\_GNUC\_PATCHLEVEL\_\_);  
#endif  
  
#ifdef \_MSC\_VER  
 printf("- Microsoft Visual C++ version: %d\n", \_MSC\_VER);  
#endif  
  
#ifdef \_\_clang\_\_  
 printf("- Clang version: %d.%d.%d\n", \_\_clang\_major\_\_, \_\_clang\_minor\_\_, \_\_clang\_patchlevel\_\_);  
#endif  
  
 /\* C規格情報 \*/  
 printf("\nC規格情報:\n");  
  
#ifdef \_\_STDC\_VERSION\_\_  
 printf("- \_\_STDC\_VERSION\_\_: %ldL\n", \_\_STDC\_VERSION\_\_);  
  
 if (\_\_STDC\_VERSION\_\_ >= 202311L)  
 {  
 printf("- 対応規格: C23\n");  
 }  
 else if (\_\_STDC\_VERSION\_\_ >= 201710L)  
 {  
 printf("- 対応規格: C17 (C18)\n");  
 }  
 else if (\_\_STDC\_VERSION\_\_ >= 201112L)  
 {  
 printf("- 対応規格: C11\n");  
 }  
 else if (\_\_STDC\_VERSION\_\_ >= 199901L)  
 {  
 printf("- 対応規格: C99\n");  
 }  
 else  
 {  
 printf("- 対応規格: C90 (ANSI C)\n");  
 }  
#else  
 printf("- 対応規格: C90 (ANSI C) または未定義\n");  
#endif  
  
 /\* システム情報 \*/  
 printf("\nシステム情報:\n");  
  
 printf("- int型サイズ: %lu bytes\n", (unsigned long)sizeof(int));  
 printf("- long型サイズ: %lu bytes\n", (unsigned long)sizeof(long));  
 printf("- ポインタサイズ: %lu bytes\n", (unsigned long)sizeof(void \*));  
 printf("- char型範囲: %d 〜 %d\n", CHAR\_MIN, CHAR\_MAX);  
 printf("- int型範囲: %d 〜 %d\n", INT\_MIN, INT\_MAX);  
}  
  
/\*  
 \* 基本的な計算テスト  
 \*/  
void test\_basic\_operations(void)  
{  
 int a = 10;  
 int b = 3;  
 int result;  
  
 printf("\n=== 基本演算テスト ===\n");  
  
 result = a + b;  
 printf("%d + %d = %d\n", a, b, result);  
  
 result = a - b;  
 printf("%d - %d = %d\n", a, b, result);  
  
 result = a \* b;  
 printf("%d \* %d = %d\n", a, b, result);  
  
 result = a / b;  
 printf("%d / %d = %d\n", a, b, result);  
  
 result = a % b;  
 printf("%d %% %d = %d\n", a, b, result);  
  
 printf("基本演算テスト: 正常完了\n");  
}  
  
/\*  
 \* メイン関数  
 \*/  
int main(void)  
{  
 printf("C言語開発環境確認プログラム\n");  
 printf("コンパイル日時: %s %s\n\n", \_\_DATE\_\_, \_\_TIME\_\_);  
  
 /\* 環境情報の表示 \*/  
 print\_environment\_info();  
  
 /\* 基本演算のテスト \*/  
 test\_basic\_operations();  
  
 printf("\n=== 環境確認完了 ===\n");  
 printf("この出力が正常に表示されれば、開発環境は正しく設定されています。\n");  
  
 return 0;  
}