# 第2章 基本文法・Hello World

## 対応C規格

* **主要対象:** C90
* **学習内容:** プログラムの基本構造、main関数、printf関数

## 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言語プログラムの基本構造を理解する
* main関数の役割を理解する
* printf関数を使った文字列出力ができる
* コメントの書き方を覚える
* 基本的なエスケープシーケンスを使える

## 概要と詳細

### プログラムの基本構造

C言語のプログラムは以下の基本構造を持ちます：

#include <stdio.h> /\* ヘッダーファイルのインクルード \*/  
  
int main(void) /\* メイン関数の定義 \*/  
{  
 printf("Hello, World!\n"); /\* 処理文 \*/  
 return 0; /\* 戻り値 \*/  
}

#### 各部分の詳しい説明

1. **プリプロセッサ指令** (#include <stdio.h>)
   * プログラムの「準備」をする行です
   * stdio.h（Standard Input/Output Header）は「標準入出力ヘッダー」という意味
   * printfのような入出力関数を使うために必要です
   * 例えるなら、料理を始める前に必要な道具を準備するようなものです
2. **main関数** (int main(void))
   * C言語プログラムの「心臓部」です
   * プログラムは必ずここから実行が始まります
   * intは「整数を返す」という意味
   * voidは「引数を受け取らない」という意味
   * どんなに大きなプログラムでも、main関数は1つだけです
3. **関数本体**（{ と } で囲まれた部分）
   * 実際の処理を書く場所です
   * {（開き波括弧）で始まり、}（閉じ波括弧）で終わります
   * この中に書かれた命令が順番に実行されます
   * インデント（字下げ）をして読みやすくするのが慣習です
4. **処理文** (printf("Hello, World!\n");)
   * 実際に何かをする命令です
   * printfは画面に文字を表示する関数
   * 文の最後には必ず;（セミコロン）を付けます
   * これを忘れるとコンパイルエラーになります
5. **return文** (return 0;)
   * プログラムの終了を示します
   * 0は「正常終了」を意味する特別な数字
   * エラーが発生した場合は0以外の数字を返すことがあります

#### なぜこの構造なのか？

初心者の方は「なぜこんなに複雑なの？」と思うかもしれません。実は、この構造には理由があります：

* **#include**: 車輪の再発明を避けるため
  + すでに作られた便利な機能を使えるようにします
  + printf関数も誰かが作ってくれた機能です
* **main関数**: プログラムの開始点を明確にするため
  + コンピュータがどこから実行すればよいか迷わないようにします
* **波括弧{}**: プログラムの範囲を明確にするため
  + どこからどこまでがmain関数なのかを示します
* **セミコロン;**: 文の終わりを明確にするため
  + 日本語の「。」のような役割です

### main関数の詳細

main関数はプログラムの**エントリーポイント**（実行開始点）です。野球で言えばピッチャーマウンド、演劇で言えば主役が最初に登場する場所のようなものです。

#### main関数の2つの形式

C言語のmain関数には主に2つの書き方があります：

**1. 基本形式（引数なし）:**

int main(void)  
{  
 /\* プログラムの処理 \*/  
 return 0;  
}

この形式は最もシンプルで、初心者の学習に適しています。voidは「何も受け取らない」という意味です。

**2. コマンドライン引数を受け取る形式:**

int main(int argc, char \*argv[])  
{  
 /\* プログラムの処理 \*/  
 return 0;  
}

この形式は、プログラム実行時に追加情報を受け取りたい場合に使います。例えば：

./myprogram file.txt -v

このように実行すると、file.txtや-vという情報をプログラム内で使えます。 （詳しくは後の章で学習します）

#### 戻り値の意味と重要性

main関数のreturnは、プログラムがどのように終了したかをOSに伝える重要な役割があります：

* **0**: 「すべて正常に終わりました」という合図
  + 成功を示す世界共通の値です
  + プログラムが期待通りに動作したことを意味します
* **0以外**: 「何か問題がありました」という合図
  + 1, 2, 3…など、異なる数字で異なるエラーを表現できます
  + 例：1 = ファイルが見つからない、2 = メモリ不足など

#### なぜ戻り値が重要なのか？

プログラムは単独で動くだけでなく、他のプログラムと連携することがあります：

# プログラムAが成功したらプログラムBを実行  
./programA && ./programB  
  
# プログラムCが失敗したらエラーメッセージを表示  
./programC || echo "エラーが発生しました"

このような連携を可能にするのが、main関数の戻り値です。

#### よくある疑問

**Q: なぜintなの？** A: 戻り値として整数（integer）を返すためです。成功/失敗を数字で表現します。

**Q: void main()と書いている例を見たけど？** A: 古い書き方で、現在は推奨されません。必ずint main()を使いましょう。

**Q: return 0;を忘れたらどうなる？** A: C99以降では自動的に0が返されますが、明示的に書くことを推奨します。

### printf関数

printf関数は画面に文字を表示するための関数です。「print formatted（フォーマット付き出力）」の略で、単純な文字列だけでなく、数値や変数の値も表示できる強力な機能です。

#### 基本的な使い方

最もシンプルな使い方から始めましょう：

printf("Hello, World!\n");

この1行で「Hello, World!」と表示して改行します。\nは改行を意味する特殊な記号です。

#### なぜprintf関数を使うのか？

コンピュータとの対話には出力が欠かせません： - 計算結果を確認したい - プログラムの動作状況を知りたい - エラーメッセージを表示したい

これらすべてにprintf関数を使います。

#### フォーマット指定子（変数の値を表示する魔法）

printf関数の真の力は、変数の値を表示できることです。%記号を使った「フォーマット指定子」がその鍵です：

| 指定子 | データ型 | 説明 | 例 |
| --- | --- | --- | --- |
| %d | int | 整数（decimal） | printf("年齢は%d歳です", 20); → 年齢は20歳です |
| %c | char | 1文字（character） | printf("成績は%cです", 'A'); → 成績はAです |
| %s | char\* | 文字列（string） | printf("名前は%sです", "太郎"); → 名前は太郎です |
| %f | double | 小数（float） | printf("円周率は%fです", 3.14); → 円周率は3.140000です |

#### 複数の値を同時に表示

printf関数では、複数の値を一度に表示できます：

int age = 20;  
double height = 170.5;  
printf("年齢: %d歳、身長: %.1fcm\n", age, height);  
/\* 出力: 年齢: 20歳、身長: 170.5cm \*/

%.1fは「小数点以下1桁まで表示」という意味です。

#### エスケープシーケンス（特殊文字の表現）

画面に表示できない特殊な文字を表現するための記号です。\（バックスラッシュ）で始まります：

| シーケンス | 意味 | 使用例と効果 |
| --- | --- | --- |
| \n | 改行（newline） | printf("1行目\n2行目"); → 2行に分けて表示 |
| \t | タブ（tab） | printf("名前\t年齢"); → 名前と年齢の間に大きな空白 |
| \" | ダブルクォート | printf("彼は\"天才\"だ"); → 彼は”天才”だ |
| \\ | バックスラッシュ | printf("C:\\Program Files"); → C:Files |

#### よくある使用例

/\* 基本的な挨拶 \*/  
printf("こんにちは！\n");  
  
/\* 計算結果の表示 \*/  
int a = 10, b = 20;  
printf("%d + %d = %d\n", a, b, a + b); /\* 10 + 20 = 30 \*/  
  
/\* 表形式での表示 \*/  
printf("商品名\t\t価格\t個数\n");  
printf("りんご\t\t100円\t3個\n");  
printf("みかん\t\t80円\t5個\n");  
  
/\* エラーメッセージ \*/  
printf("エラー: ファイル \"%s\" が見つかりません。\n", "data.txt");

#### 初心者がよく間違えるポイント

1. **%の数と値の数が一致しない**

* /\* NG: %dが2つあるのに値が1つしかない \*/  
  printf("%d + %d = ", 10); /\* エラーまたは予期しない動作 \*/  
    
  /\* OK: 正しい書き方 \*/  
  printf("%d + %d = %d", 10, 20, 30);

1. \*\*\*

* printf("Hello");  
  printf("World"); /\* HelloWorld と続けて表示される \*/  
    
  /\* 改行したい場合 \*/  
  printf("Hello\n");  
  printf("World\n");

1. **%と変数の型が一致しない**

* double pi = 3.14;  
  /\* NG: doubleに%dを使用 \*/  
  printf("円周率は%dです", pi); /\* おかしな値が表示される \*/  
    
  /\* OK: doubleには%f \*/  
  printf("円周率は%fです", pi);

### コメント

コメントは、プログラムに説明を追加するための重要な機能です。プログラムを書いた本人でも、時間が経つと「これは何のためのコードだっけ？」と忘れてしまうことがあります。コメントはそんな時に役立つ「メモ書き」です。

#### なぜコメントが重要なのか？

1. **将来の自分のため**
   * 3ヶ月後の自分は他人と同じです
   * 何を考えてこのコードを書いたか思い出せます
2. **チームメンバーのため**
   * 他の人があなたのコードを理解しやすくなります
   * 協力して開発する際には必須です
3. **デバッグのため**
   * 問題が起きた時に、意図を理解しやすくなります
   * コードの一部を一時的に無効にする際にも使えます

#### C90スタイル（伝統的な書き方）

/\* これは複数行にわたる  
 コメントです。  
 何行でも書くことができます。 \*/  
  
/\* 1行コメント \*/  
  
int age = 20; /\* 行の途中からでもコメントが書けます \*/

#### C99以降のスタイル（新しい書き方）

// これは1行コメントです（C99以降）  
// 複数行書きたい場合は  
// このように各行に//を付けます  
  
int age = 20; // 行末コメントも簡単に書けます

**重要:** この章ではC90準拠のため、/\* \*/ スタイルを使用します。

#### 良いコメントの書き方

**良い例:**

/\* 消費税込みの価格を計算する \*/  
int price\_with\_tax = price \* 1.1;  
  
/\* ユーザーが入力した値が有効範囲内かチェック \*/  
if (value >= 0 && value <= 100) {  
 /\* 処理を実行 \*/  
}

**悪い例:**

/\* 1を足す \*/  
i = i + 1; /\* コードを見れば分かることは書かない \*/  
  
/\* ここで変数を宣言 \*/  
int x; /\* 当たり前のことは書かない \*/

#### コメントアウト（一時的にコードを無効化）

デバッグ時によく使うテクニックです：

printf("処理開始\n");  
/\* printf("デバッグ: 変数の値は%d\n", x); \*/ /\* 一時的に無効化 \*/  
printf("処理終了\n");

### 基本的なプログラム例

実際のプログラムを見てみましょう。最初は難しく感じるかもしれませんが、一つずつ理解していけば大丈夫です。

#### 例1: もっともシンプルなプログラム

#include <stdio.h>  
  
int main(void)  
{  
 printf("Hello, World!\n");  
 return 0;  
}

**このプログラムのポイント:** - たった6行で完全に動作するプログラムです - 「Hello, World!」と表示して終了します - プログラミングの第一歩として世界中で使われている例です

#### 例2: 複数行の出力

#include <stdio.h>  
  
int main(void)  
{  
 printf("C言語学習教材\n");  
 printf("基本文法・Hello World\n");  
 printf("プログラムが正常に動作しています！\n");  
 return 0;  
}

**このプログラムのポイント:** - printf関数を3回使って3行表示しています - 各printf関数は上から順番に実行されます - \nがあるので、それぞれ改行されて表示されます

#### 例3: エスケープシーケンスの活用

#include <stdio.h>  
  
int main(void)  
{  
 printf("タブ区切り:\tアイテム1\tアイテム2\n");  
 printf("引用符の表示: \"Hello, World!\"\n");  
 printf("パス表示: C:\\Program Files\\MyApp\n");  
 return 0;  
}

**このプログラムのポイント:** - \tでタブ（大きな空白）を入れて整列させています - \"で文字列の中にダブルクォートを含めています - \\でバックスラッシュ自体を表示しています（Windowsのパス表示でよく使います）

#### 実行結果の例

上記のプログラムを実行すると、以下のような出力が得られます：

タブ区切り: アイテム1 アイテム2  
引用符の表示: "Hello, World!"  
パス表示: C:\Program Files\MyApp

#### 初心者へのアドバイス

1. **まずは写経から**
   * 最初はサンプルコードをそのまま入力してみましょう
   * タイピングすることで、構文に慣れることができます
2. **少しずつ変更してみる**
   * “Hello, World!”を自分の名前に変えてみる
   * printf関数を追加してみる
   * エスケープシーケンスを組み合わせてみる
3. **エラーを恐れない**
   * セミコロンを忘れたり、括弧を閉じ忘れたりするのは普通です
   * エラーメッセージをよく読んで、何が間違っているか理解しましょう

## サンプルコード

### Hello Worldプログラム

プログラムファイル: examples/hello\_world.c

もっとも基本的なC言語プログラムです。

### フォーマット出力プログラム

プログラムファイル: examples/formatted\_output.c

printf関数のさまざまな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 hello\_world.c -o hello\_world  
  
# 実行  
./hello\_world

## 演習課題

### 基礎問題

1. **Hello World**
   * 自分の名前を表示するプログラムを作成してください
2. **複数実行出力**
   * 好きな詩や歌詞を3実行以上で表示するプログラムを作成してください
3. **エスケープシーケンス**
   * タブとダブルクォートを使った表示プログラムを作成してください

### 応用問題

1. **フォーマット文字列**
   * printf関数のさまざまなフォーマット指定子を試してください
2. **プログラム構造の理解**
   * main関数の戻り値を変更して動作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  
  
# 特定のプログラムをコンパイル  
make hello\_world  
  
# 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

* コメントは /\* \*/ 形式のみ
* 変数宣言は関数やブロックの先頭で実行う
* forループ内での変数宣言は不可

### C99以降の拡張

* // 形式のコメント追加
* forループ内での変数宣言が可能
* 複合リテラル等の新機能

## 次の章へ

基本文法を理解したら、[データ型と変数](../data-types/README.md) に進んでください。

## 参考資料

* [printf関数リファレンス](https://ja.cppreference.com/w/c/io/fprintf)
* [エスケープシーケンス一覧](https://ja.cppreference.com/w/c/language/escape)
* [C90規格仕様](https://www.iso.org/standard/17782.html)

# サンプルコード

## formatted\_output.c

/\*  
 \* ファイル名: formatted\_output.c  
 \* 説明: printf関数のさまざまなフォーマット指定子を学習  
 \* 規格: C90準拠  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 /\* 基本的な文字列出力 \*/  
 printf("=== C言語フォーマット出力 ===\n");  
  
 /\* 整数の出力 \*/  
 printf("整数: %d\n", 42);  
 printf("負の整数: %d\n", -123);  
  
 /\* 文字の出力 \*/  
 printf("文字: %c\n", 'A');  
 printf("文字コード: %c\n", 65); /\* Aのアスキーコード \*/  
  
 /\* 文字列の出力 \*/  
 printf("文字列: %s\n", "Hello, World!");  
  
 /\* 浮動小数点数の出力 \*/  
 printf("浮動小数点数: %f\n", 3.14159);  
 printf("小数点以下2桁: %.2f\n", 3.14159);  
  
 /\* エスケープシーケンスの使用 \*/  
 printf("\n--- エスケープシーケンス ---\n");  
 printf("改行\nとタブ\tの使用例：\n");  
 printf("項目1\t項目2\t項目3\n");  
 printf("値A\t値B\t値C\n");  
  
 /\* 特殊文字の表示 \*/  
 printf("\n--- 特殊文字の表示 ---\n");  
 printf("ダブルクォート: \"Hello\"\n");  
 printf("バックスラッシュ: C:\\Program Files\\\n");  
  
 /\* フォーマット指定子の組み合わせ \*/  
 printf("\n--- 複合表示 ---\n");  
 printf("名前: %s, 年齢: %d歳, 身長: %.1fcm\n", "田中太郎", 25, 175.5);  
  
 return 0;  
}

## hello\_world.c

/\*  
 \* ファイル名: hello\_world.c  
 \* 説明: C言語の最も基本的なプログラム  
 \* 規格: C90準拠  
 \*/  
  
#include <stdio.h>  
  
int main(void)  
{  
 printf("Hello, World!\n");  
 return 0;  
}