# 第5章 制御構造（条件分岐） - 演習問題

# 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

## 演習問題

### 演習4-1: 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

### 演習4-2: 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

### 演習4-3: 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

### 演習4-4: 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

### 演習4-5: 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

**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

## チャレンジ問題

### チャレンジ4-6: 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1回の結果を記録して最後に表示

## 提出方法

1. 各演習問題ごとに独立したCファイルを作成
2. ファイル名は ex4\_1.c, ex4\_2.c のような形式で
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・実行確認を必ず実行う

## 期限

演習問題は次の章に進む前に完了させましょう。

# 解答例

## 解答例一覧

### 高度な解答例

#### 成績計算システム

* **ファイル**: grade\_calculator.c, grade\_calculator\_c99.c
* **学習内容**: if-else if連鎖、switch文、複合条件判定
* **ポイント**: 実用的な評価システムの実装
* **機能**: テスト点数、出席率、宿題提出状況による総合評価

## 演習問題の解答

### 演習4-1: 年齢による料金計算

* **ファイル**: ex4\_1\_ticket\_pricing.c, ex4\_1\_ticket\_pricing\_c99.c
* **学習内容**: if-else if文による段階的判定
* **ポイント**: 境界値の適切な処理

### 演習4-2: 電卓プログラム

* **ファイル**: ex4\_2\_calculator.c, ex4\_2\_calculator\_c99.c
* **学習内容**: switch文による多分岐処理
* **ポイント**: ゼロ除算エラーの処理

### 演習4-3: BMI判定プログラム

* **ファイル**: ex4\_3\_bmi\_judge.c, ex4\_3\_bmi\_judge\_c99.c
* **学習内容**: 浮動小数点数の計算と範囲判定
* **ポイント**: 単位変換と健康アドバイス

### 演習4-4: うるう年判定

* **ファイル**: ex4\_4\_leap\_year.c, ex4\_4\_leap\_year\_c99.c
* **学習内容**: 複雑な論理演算子の組み合わせ
* **ポイント**: 複合条件式の正確な実装

### 演習4-5: 成績評価システム

* **ファイル**: ex4\_5\_grade\_system.c, ex4\_5\_grade\_system\_c99.c
* **学習内容**: 条件演算子と複合条件の活用
* **ポイント**: 複数データの統合的な評価

### チャレンジ4-6: じゃんけんゲーム

* **ファイル**: ex4\_6\_janken\_game.c, ex4\_6\_janken\_game\_c99.c
* **学習内容**: ランダム数生成と複雑な勝敗判定
* **ポイント**: ゲームロジックの実装と統計分析

## 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言語の制御構造に準拠

### 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
* より現代的な制御構造の書き方

## コンパイルと実行

### 基本的なコンパイル

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

### C99版のコンパイル

gcc -Wall -Wextra -pedantic -std=c99 ex4\_1\_ticket\_pricing\_c99.c -o ex4\_1\_ticket\_pricing\_c99  
./ex4\_1\_ticket\_pricing\_c99

### Makefileを使用

make all # 全てをコンパイル  
make solutions # 解答例のみコンパイル  
make ex4\_1 # 個別コンパイル  
make run-solutions # 解答例を実行  
make clean # 実行ファイルを削除

## 学習のポイント

### 条件分岐の基本

1. **if文**: 単純な条件判定
2. **if-else文**: 2分岐の処理
3. **if-else if文**: 多段階の条件判定
4. **switch文**: 値による多分岐処理
5. **条件演算子**: 簡潔な条件分岐

### 実用的なテクニック

1. **入力検証**: ユーザー入力の安全性確保
2. **エラーハンドリング**: 適切なエラー処理
3. **境界値処理**: 範囲指定の正確な実装
4. **複合条件**: 論理演算子の効果的な使用
5. **可読性**: 理解しやすいコードの書き方

### 設計パターン

1. **段階的判定**: if-else if連鎖による階層的処理
2. **値による分岐**: switch文による効率的な分岐
3. **状態管理**: フラグ変数による状態の管理
4. **統合的判定**: 複数条件による総合的な評価

## 注意事項

* 条件式では等価演算子（==）と代入演算子（=）を混同しない
* switch文ではbreak文を忘れずに記述
* 浮動小数点数の比較では誤差を考慮
* 論理演算子の優先順位に注意

## 実用的な応用

これらの制御構造は以下のような実用的なプログラムで使用されます：

* **Webアプリケーション**: ユーザー認証、フォーム検証
* **ゲーム**: プレイヤーの行動による分岐処理
* **データ分析**: 条件による分類・集計処理
* **システム管理**: 設定による動作の制御

## 次のステップ

この章をマスターしたら、次の章に進みましょう： - [第5章: 制御構造（繰り返し）](../control-loop/) - [第6章: 配列](../arrays/) - [第7章: 文字列](../strings/) - [第8章: 関数](../functions/) ## ex4\_1\_ticket\_pricing.c

```c / ファイル名: ex4\_1\_ticket\_pricing.c \* 演習4-1: 年齢による料金計算 \* 説明: 映画館の料金システムを実装するプログラム \* 規格: C90準拠 \*/ #include <stdio.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） */ int age; int price; char* category;

printf("=== 映画館料金計算システム ===\n");  
printf("年齢を入力してください: ");  
scanf("%d", &age);  
  
/\* 入力値の検証 \*/  
if (age < 0) {  
 printf("エラー: 年齢は0以上の値を入力してください。\n");  
 return 1;  
}  
  
if (age > 120) {  
 printf("エラー: 年齢は120歳以下の値を入力してください。\n");  
 return 1;  
}  
  
/\* 年齢による料金分類 \*/  
if (age < 3) {  
 price = 0;  
 category = "幼児";  
} else if (age >= 3 && age <= 12) {  
 price = 800;  
 category = "子供";  
} else if (age >= 13 && age <= 18) {  
 price = 1200;  
 category = "学生";  
} else if (age >= 19 && age <= 65) {  
 price = 1800;  
 category = "大人";  
} else {  
 price = 1200;  
 category = "シニア";  
}  
  
/\* 結果の表示 \*/  
printf("\n=== 料金計算結果 ===\n");  
printf("年齢: %d歳\n", age);  
printf("区分: %s\n", category);  
  
if (price == 0) {  
 printf("料金: 無料\n");  
} else {  
 printf("料金: %d円\n", price);  
}  
  
/\* 特別なメッセージ \*/  
if (age < 3) {  
 printf("お子様は保護者の同伴が必要です。\n");  
} else if (age >= 66) {  
 printf("シニア割引が適用されました。\n");  
} else if (age >= 13 && age <= 18) {  
 printf("学生証の提示をお願いします。\n");  
}  
  
printf("\nご利用ありがとうございます！\n");  
  
return 0;

}

/\* 学習ポイント: 1. if-else if文による段階的な条件判定 2. 境界値の処理（<=, >=の使い分け） 3. 入力値の検証とエラーハンドリング 4. 論理演算子（&&）による範囲指定 5. 変数の適切な初期化と使用

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - 境界値（3歳、12歳、18歳、65歳）を正確に処理 - ユーザーフレンドリーなエラーメッセージ - 明確な結果表示 \*/```

## ex4\_2\_calculator.c

```c / ファイル名: ex4\_2\_calculator.c \* 演習4-2: 電卓プログラム \* 説明: 四則演算ができる簡単な電卓プログラム \* 規格: C90準拠 \*/ #include <stdio.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） \*/ double num1, num2, result; char operator; int valid\_operator;

printf("=== 簡単電卓プログラム ===\n");  
  
/\* 第一の数値入力 \*/  
printf("最初の数値を入力してください: ");  
scanf("%lf", &num1);  
  
/\* 演算子の入力 \*/  
printf("演算子を入力してください (+, -, \*, /): ");  
scanf(" %c", &operator); /\* 空白文字を読み飛ばすため、%cの前に空白 \*/  
  
/\* 第二の数値入力 \*/  
printf("二番目の数値を入力してください: ");  
scanf("%lf", &num2);  
  
/\* 演算子の妥当性チェック \*/  
valid\_operator = 1;  
  
/\* switch文による演算の実行 \*/  
switch (operator) {  
 case '+':  
 result = num1 + num2;  
 printf("%.2f + %.2f = %.2f\n", num1, num2, result);  
 break;  
   
 case '-':  
 result = num1 - num2;  
 printf("%.2f - %.2f = %.2f\n", num1, num2, result);  
 break;  
   
 case '\*':  
 result = num1 \* num2;  
 printf("%.2f \* %.2f = %.2f\n", num1, num2, result);  
 break;  
   
 case '/':  
 /\* ゼロ除算チェック \*/  
 if (num2 == 0.0) {  
 printf("エラー: ゼロで割ることはできません。\n");  
 return 1;  
 } else {  
 result = num1 / num2;  
 printf("%.2f / %.2f = %.2f\n", num1, num2, result);  
 }  
 break;  
   
 default:  
 printf("エラー: 無効な演算子です。(+, -, \*, /) のいずれかを使用してください。\n");  
 valid\_operator = 0;  
 break;  
}  
  
/\* 演算が成功した場合の追加情報 \*/  
if (valid\_operator && operator != '/') {  
 printf("\n=== 計算完了 ===\n");  
   
 /\* 結果の分析 \*/  
 if (result > 0) {  
 printf("結果は正の数です。\n");  
 } else if (result < 0) {  
 printf("結果は負の数です。\n");  
 } else {  
 printf("結果はゼロです。\n");  
 }  
   
 /\* 整数かどうかのチェック \*/  
 if (result == (int)result) {  
 printf("結果は整数です。\n");  
 } else {  
 printf("結果は小数です。\n");  
 }  
} else if (valid\_operator && operator == '/') {  
 printf("\n=== 除算完了 ===\n");  
   
 /\* 除算の特別な処理 \*/  
 if (result == 1.0) {  
 printf("割る数と割られる数が同じです。\n");  
 } else if (result > 1.0) {  
 printf("割られる数の方が大きいです。\n");  
 } else {  
 printf("割る数の方が大きいです。\n");  
 }  
}  
  
printf("電卓プログラムを終了します。\n");  
  
return 0;

}

/\* 学習ポイント: 1. switch文による多分岐処理 2. ゼロ除算のエラーハンドリング 3. 浮動小数点数の比較 4. break文の重要性 5. default句によるエラー処理

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - scanf()での文字入力時の注意点（空白文字の処理） - 浮動小数点数の比較（== 0.0の使用） - switch文での各caseの適切な処理 - ユーザーフレンドリーなエラーメッセージ \*/```

## ex4\_3\_bmi\_judge.c

```c / ファイル名: ex4\_3\_bmi\_judge.c \* 演習4-3: BMI判定プログラム \* 説明: 身長と体重からBMIを計算し体型判定を行う \* 規格: C90準拠 \*/ #include <stdio.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） */ double height\_cm, height\_m, weight, bmi; char* category; char \*advice;

printf("=== BMI判定プログラム ===\n");  
  
/\* 身長の入力（センチメートル） \*/  
printf("身長を入力してください（cm）: ");  
scanf("%lf", &height\_cm);  
  
/\* 体重の入力 \*/  
printf("体重を入力してください（kg）: ");  
scanf("%lf", &weight);  
  
/\* 入力値の検証 \*/  
if (height\_cm <= 0 || height\_cm > 300) {  
 printf("エラー: 身長は1cm〜300cmの範囲で入力してください。\n");  
 return 1;  
}  
  
if (weight <= 0 || weight > 500) {  
 printf("エラー: 体重は1kg〜500kgの範囲で入力してください。\n");  
 return 1;  
}  
  
/\* 身長をメートルに変換 \*/  
height\_m = height\_cm / 100.0;  
  
/\* BMI計算 \*/  
bmi = weight / (height\_m \* height\_m);  
  
/\* BMI判定 \*/  
if (bmi < 18.5) {  
 category = "低体重";  
 advice = "栄養バランスの良い食事を心がけ、適度な運動で健康的に体重を増やしましょう。";  
} else if (bmi >= 18.5 && bmi < 25.0) {  
 category = "標準体重";  
 advice = "理想的な体重です。現在の生活習慣を維持しましょう。";  
} else if (bmi >= 25.0 && bmi < 30.0) {  
 category = "肥満度1";  
 advice = "軽度の肥満です。食事制限と有酸素運動を始めることをお勧めします。";  
} else {  
 category = "肥満度2";  
 advice = "高度な肥満です。医師に相談し、本格的な減量プログラムを検討してください。";  
}  
  
/\* 結果の表示 \*/  
printf("\n=== BMI計算結果 ===\n");  
printf("身長: %.1f cm (%.2f m)\n", height\_cm, height\_m);  
printf("体重: %.1f kg\n", weight);  
printf("BMI: %.2f\n", bmi);  
printf("判定: %s\n", category);  
printf("\nアドバイス: %s\n", advice);  
  
/\* 追加の健康情報 \*/  
printf("\n=== 健康情報 ===\n");  
  
if (bmi < 18.5) {  
 printf("・BMI 18.5未満は痩せすぎの可能性があります\n");  
 printf("・免疫力低下や骨密度減少のリスクがあります\n");  
} else if (bmi >= 18.5 && bmi < 25.0) {  
 printf("・BMI 18.5-24.9は標準的な範囲です\n");  
 printf("・生活習慣病のリスクが最も低い範囲です\n");  
} else if (bmi >= 25.0 && bmi < 30.0) {  
 printf("・BMI 25.0-29.9は軽度肥満の範囲です\n");  
 printf("・糖尿病や高血圧のリスクが上昇します\n");  
} else {  
 printf("・BMI 30.0以上は高度肥満の範囲です\n");  
 printf("・心血管疾患のリスクが大幅に増加します\n");  
}  
  
/\* 理想体重の計算と表示 \*/  
if (bmi < 18.5 || bmi >= 25.0) {  
 double ideal\_weight\_min = 18.5 \* (height\_m \* height\_m);  
 double ideal\_weight\_max = 24.9 \* (height\_m \* height\_m);  
   
 printf("\n=== 理想体重の目安 ===\n");  
 printf("あなたの理想体重: %.1f kg 〜 %.1f kg\n",   
 ideal\_weight\_min, ideal\_weight\_max);  
   
 if (weight < ideal\_weight\_min) {  
 printf("目標: %.1f kg の増量\n", ideal\_weight\_min - weight);  
 } else if (weight > ideal\_weight\_max) {  
 printf("目標: %.1f kg の減量\n", weight - ideal\_weight\_max);  
 }  
}  
  
printf("\n※この判定は一般的な目安です。詳しくは医師にご相談ください。\n");  
  
return 0;

}

/\* 学習ポイント: 1. 浮動小数点数を使った計算 2. if-else if文による範囲判定 3. 単位変換（cm→m） 4. 論理演算子による範囲指定 5. 計算結果を使った追加処理

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - BMI計算式の正確な実装 - 境界値（18.5, 25.0, 30.0）の適切な処理 - ユーザーフレンドリーな結果表示 - 入力値の妥当性チェック \*/```

## ex4\_4\_leap\_year.c

```c / ファイル名: ex4\_4\_leap\_year.c \* 演習4-4: うるう年判定 \* 説明: 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 \* 規格: C90準拠 \*/ #include <stdio.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） \*/ int year; int is\_leap\_year;

printf("=== うるう年判定プログラム ===\n");  
printf("西暦年を入力してください: ");  
scanf("%d", &year);  
  
/\* 入力値の検証 \*/  
if (year < 1) {  
 printf("エラー: 西暦1年以降を入力してください。\n");  
 return 1;  
}  
  
if (year > 3000) {  
 printf("エラー: 西暦3000年以下を入力してください。\n");  
 return 1;  
}  
  
/\* うるう年判定のロジック \*/  
/\* 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 \*/  
  
if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {  
 is\_leap\_year = 1;  
} else {  
 is\_leap\_year = 0;  
}  
  
/\* 結果の表示 \*/  
printf("\n=== 判定結果 ===\n");  
printf("入力年: %d年\n", year);  
  
if (is\_leap\_year) {  
 printf("判定: うるう年です\n");  
 printf("2月は29日まであります。\n");  
} else {  
 printf("判定: 平年です\n");  
 printf("2月は28日まであります。\n");  
}  
  
/\* 詳細な判定理由の表示 \*/  
printf("\n=== 判定理由 ===\n");  
  
if (year % 4 != 0) {  
 printf("4で割り切れないため、平年です。\n");  
} else if (year % 100 != 0) {  
 printf("4で割り切れ、かつ100で割り切れないため、うるう年です。\n");  
} else if (year % 400 != 0) {  
 printf("100で割り切れ、かつ400で割り切れないため、平年です。\n");  
} else {  
 printf("400で割り切れるため、うるう年です。\n");  
}  
  
/\* 歴史的な情報 \*/  
printf("\n=== 豆知識 ===\n");  
  
if (year == 2000) {  
 printf("2000年は特別なうるう年でした（400で割り切れる）。\n");  
} else if (year == 1900) {  
 printf("1900年は100で割り切れるが400で割り切れないため平年でした。\n");  
} else if (year == 2100) {  
 printf("2100年も100で割り切れるが400で割り切れないため平年になります。\n");  
}  
  
/\* グレゴリオ暦の説明 \*/  
if (year >= 1582) {  
 printf("グレゴリオ暦（現在の暦）での判定です。\n");  
} else {  
 printf("注意: 1582年以前はユリウス暦が使われていました。\n");  
}  
  
/\* 次のうるう年/平年の計算 \*/  
printf("\n=== 次のうるう年 ===\n");  
  
if (is\_leap\_year) {  
 /\* 現在がうるう年の場合、次のうるう年を探す \*/  
 int next\_year;  
 for (next\_year = year + 1; next\_year <= year + 8; next\_year++) {  
 if ((next\_year % 4 == 0 && next\_year % 100 != 0) || (next\_year % 400 == 0)) {  
 printf("次のうるう年: %d年\n", next\_year);  
 break;  
 }  
 }  
} else {  
 /\* 現在が平年の場合、次のうるう年を探す \*/  
 int next\_year;  
 for (next\_year = year + 1; next\_year <= year + 4; next\_year++) {  
 if ((next\_year % 4 == 0 && next\_year % 100 != 0) || (next\_year % 400 == 0)) {  
 printf("次のうるう年: %d年\n", next\_year);  
 break;  
 }  
 }  
}  
  
return 0;

}

/\* 学習ポイント: 1. 複雑な論理演算子の組み合わせ 2. 剰余演算子（%）による割り切れる判定 3. 条件の優先順位と論理構造 4. 複合条件式の読みやすい書き方 5. ループを使った条件検索

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - うるう年判定の複雑なルールを正確に実装 - 論理演算子（&&, ||）の適切な使用 - 条件式を段階的に説明 - 歴史的背景も含めた教育的な内容 \*/```

## ex4\_5\_grade\_system.c

```c / ファイル名: ex4\_5\_grade\_system.c \* 演習4-5: 成績評価システム \* 説明: 3科目の点数から平均点を計算し総合評価を行う \* 規格: C90準拠 \*/ #include <stdio.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） */ int japanese, math, english; int total; double average; char grade; char* evaluation; int all\_passed; char \*result;

printf("=== 成績評価システム ===\n");  
  
/\* 各科目の点数入力 \*/  
printf("国語の点数を入力してください (0-100): ");  
scanf("%d", &japanese);  
  
printf("数学の点数を入力してください (0-100): ");  
scanf("%d", &math);  
  
printf("英語の点数を入力してください (0-100): ");  
scanf("%d", &english);  
  
/\* 入力値の検証 \*/  
if (japanese < 0 || japanese > 100) {  
 printf("エラー: 国語の点数は0-100の範囲で入力してください。\n");  
 return 1;  
}  
  
if (math < 0 || math > 100) {  
 printf("エラー: 数学の点数は0-100の範囲で入力してください。\n");  
 return 1;  
}  
  
if (english < 0 || english > 100) {  
 printf("エラー: 英語の点数は0-100の範囲で入力してください。\n");  
 return 1;  
}  
  
/\* 合計点と平均点の計算 \*/  
total = japanese + math + english;  
average = (double)total / 3.0; /\* 整数演算を避けるため型キャスト \*/  
  
/\* 平均点による評価 \*/  
if (average >= 90) {  
 grade = 'A';  
 evaluation = "優秀";  
} else if (average >= 80) {  
 grade = 'B';  
 evaluation = "良好";  
} else if (average >= 70) {  
 grade = 'C';  
 evaluation = "普通";  
} else if (average >= 60) {  
 grade = 'D';  
 evaluation = "可";  
} else {  
 grade = 'F';  
 evaluation = "不可";  
}  
  
/\* 全科目60点以上の確認 \*/  
all\_passed = (japanese >= 60 && math >= 60 && english >= 60) ? 1 : 0;  
  
/\* 条件演算子を使った簡潔な表示 \*/  
result = all\_passed ? "合格" : "不合格";  
  
/\* 結果の表示 \*/  
printf("\n=== 成績評価結果 ===\n");  
printf("国語: %d点\n", japanese);  
printf("数学: %d点\n", math);  
printf("英語: %d点\n", english);  
printf("--------------------\n");  
printf("合計点: %d点\n", total);  
printf("平均点: %.2f点\n", average);  
printf("評価: %c (%s)\n", grade, evaluation);  
printf("判定: %s\n", result);  
  
/\* 詳細な分析 \*/  
printf("\n=== 詳細分析 ===\n");  
  
/\* 各科目の個別評価 \*/  
printf("科目別評価:\n");  
printf("・国語: %s\n", (japanese >= 60) ? "合格" : "不合格");  
printf("・数学: %s\n", (math >= 60) ? "合格" : "不合格");  
printf("・英語: %s\n", (english >= 60) ? "合格" : "不合格");  
  
/\* 最高点と最低点の科目 \*/  
if (japanese >= math && japanese >= english) {  
 printf("最高得点科目: 国語 (%d点)\n", japanese);  
} else if (math >= japanese && math >= english) {  
 printf("最高得点科目: 数学 (%d点)\n", math);  
} else {  
 printf("最高得点科目: 英語 (%d点)\n", english);  
}  
  
if (japanese <= math && japanese <= english) {  
 printf("最低得点科目: 国語 (%d点)\n", japanese);  
} else if (math <= japanese && math <= english) {  
 printf("最低得点科目: 数学 (%d点)\n", math);  
} else {  
 printf("最低得点科目: 英語 (%d点)\n", english);  
}  
  
/\* 改善アドバイス \*/  
printf("\n=== 改善アドバイス ===\n");  
  
if (all\_passed) {  
 printf("全科目合格おめでとうございます！\n");  
 if (average >= 90) {  
 printf("優秀な成績です。この調子で頑張ってください。\n");  
 } else if (average >= 80) {  
 printf("良い成績です。さらなる向上を目指しましょう。\n");  
 } else {  
 printf("全科目クリアしました。より高い点数を目指しましょう。\n");  
 }  
} else {  
 printf("不合格科目があります。以下の科目の復習をしましょう:\n");  
 if (japanese < 60) {  
 printf("・国語: %d点 (不足 %d点)\n", japanese, 60 - japanese);  
 }  
 if (math < 60) {  
 printf("・数学: %d点 (不足 %d点)\n", math, 60 - math);  
 }  
 if (english < 60) {  
 printf("・英語: %d点 (不足 %d点)\n", english, 60 - english);  
 }  
}  
  
/\* 次回の目標設定 \*/  
printf("\n=== 次回の目標 ===\n");  
if (grade == 'A') {  
 printf("目標: A評価の維持\n");  
} else if (grade == 'B') {  
 printf("目標: A評価（平均90点以上）\n");  
} else if (grade == 'C') {  
 printf("目標: B評価（平均80点以上）\n");  
} else if (grade == 'D') {  
 printf("目標: C評価（平均70点以上）\n");  
} else {  
 printf("目標: まずは全科目60点以上\n");  
}  
  
return 0;

}

/\* 学習ポイント: 1. 複数の変数を使った計算処理 2. if-else if文による段階的評価 3. 条件演算子（?:）の実用的な使用 4. 論理演算子（&&）による複合条件 5. 型キャスト（int→double）による精密な計算

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - 整数演算を避けるための型キャスト - 条件演算子による簡潔な条件分岐 - 複数条件の組み合わせによる判定 - ユーザーフレンドリーな結果表示 \*/```

## ex4\_6\_janken\_game.c

```c / ファイル名: ex4\_6\_janken\_game.c \* チャレンジ4-6: じゃんけんゲーム \* 説明: コンピュータとじゃんけんをする3回勝負のプログラム \* 規格: C90準拠 \*/ #include <stdio.h> #include <stdlib.h> #include <time.h>

int main(void) { /\* 変数の宣言（C90では先頭で宣言） */ int user\_choice, computer\_choice; int user\_wins, computer\_wins, draws; int round; char* hands[3] = {“グー”, “チョキ”, “パー”}; char \*result\_message;

/\* 勝利数の初期化 \*/  
user\_wins = 0;  
computer\_wins = 0;  
draws = 0;  
  
/\* 乱数の初期化 \*/  
srand((unsigned int)time(NULL));  
  
printf("=== じゃんけんゲーム（3回勝負） ===\n");  
printf("1: グー, 2: チョキ, 3: パー\n");  
printf("さあ、始めましょう！\n\n");  
  
/\* 3回勝負のループ \*/  
for (round = 1; round <= 3; round++) {  
 printf("--- 第%d回戦 ---\n", round);  
   
 /\* ユーザーの手の入力 \*/  
 do {  
 printf("あなたの手を選んでください (1-3): ");  
 scanf("%d", &user\_choice);  
   
 if (user\_choice < 1 || user\_choice > 3) {  
 printf("エラー: 1〜3の数字を入力してください。\n");  
 }  
 } while (user\_choice < 1 || user\_choice > 3);  
   
 /\* コンピュータの手をランダムに決定 \*/  
 computer\_choice = rand() % 3 + 1;  
   
 /\* 手の表示 \*/  
 printf("あなた: %s\n", hands[user\_choice - 1]);  
 printf("コンピュータ: %s\n", hands[computer\_choice - 1]);  
   
 /\* 勝敗判定 \*/  
 if (user\_choice == computer\_choice) {  
 printf("結果: あいこ\n");  
 result\_message = "あいこ";  
 draws++;  
 } else if ((user\_choice == 1 && computer\_choice == 2) || /\* グー vs チョキ \*/  
 (user\_choice == 2 && computer\_choice == 3) || /\* チョキ vs パー \*/  
 (user\_choice == 3 && computer\_choice == 1)) { /\* パー vs グー \*/  
 printf("結果: あなたの勝ち！\n");  
 result\_message = "勝利";  
 user\_wins++;  
 } else {  
 printf("結果: コンピュータの勝ち！\n");  
 result\_message = "敗北";  
 computer\_wins++;  
 }  
   
 /\* 現在の戦績表示 \*/  
 printf("現在の戦績 - あなた: %d勝, コンピュータ: %d勝, あいこ: %d回\n",   
 user\_wins, computer\_wins, draws);  
 printf("\n");  
}  
  
/\* 最終結果の表示 \*/  
printf("=== 最終結果 ===\n");  
printf("3回戦の結果:\n");  
printf("あなた: %d勝\n", user\_wins);  
printf("コンピュータ: %d勝\n", computer\_wins);  
printf("あいこ: %d回\n", draws);  
printf("-----------------\n");  
  
/\* 総合勝者の判定 \*/  
if (user\_wins > computer\_wins) {  
 printf("総合結果: あなたの勝利！\n");  
 printf("おめでとうございます！素晴らしい戦いでした。\n");  
   
 if (user\_wins == 3) {  
 printf("完全勝利です！パーフェクトゲーム達成！\n");  
 } else if (user\_wins == 2 && draws == 1) {  
 printf("2勝1分けの安定した勝利です。\n");  
 }  
} else if (computer\_wins > user\_wins) {  
 printf("総合結果: コンピュータの勝利！\n");  
 printf("残念！次回はきっと勝てますよ。\n");  
   
 if (computer\_wins == 3) {  
 printf("完敗でした。次回に期待しましょう！\n");  
 }  
} else {  
 printf("総合結果: 引き分け！\n");  
 printf("互角の戦いでした。良いゲームでしたね。\n");  
}  
  
/\* 詳細な分析 \*/  
printf("\n=== ゲーム分析 ===\n");  
  
/\* 勝率の計算 \*/  
if (user\_wins + computer\_wins > 0) {  
 double win\_rate = (double)user\_wins / (user\_wins + computer\_wins) \* 100.0;  
 printf("あなたの勝率: %.1f%%\n", win\_rate);  
   
 if (win\_rate >= 75.0) {  
 printf("評価: 優秀な戦績です！\n");  
 } else if (win\_rate >= 50.0) {  
 printf("評価: まずまずの戦績です。\n");  
 } else {  
 printf("評価: 次回頑張りましょう！\n");  
 }  
}  
  
/\* あいこの回数による分析 \*/  
if (draws == 0) {  
 printf("あいこなしの白熱した戦いでした。\n");  
} else if (draws == 1) {  
 printf("1回のあいこがありました。\n");  
} else if (draws == 2) {  
 printf("2回のあいこがありました。運命的な戦いでしたね。\n");  
} else {  
 printf("全てあいこ！これは珍しい結果です。\n");  
}  
  
/\* ランダム性の説明 \*/  
printf("\n=== 豆知識 ===\n");  
printf("・コンピュータの手は完全にランダムです\n");  
printf("・じゃんけんの確率論では、各手が出る確率は1/3です\n");  
printf("・人間は無意識にパターンを作りがちですが、\n");  
printf(" 本当のランダム性には勝てません\n");  
  
printf("\nゲーム終了！また遊んでくださいね。\n");  
  
return 0;

}

/\* 学習ポイント: 1. ランダム数の生成（srand, rand） 2. 複雑な条件分岐による勝敗判定 3. ループを使った繰り返し処理 4. 配列を使った文字列の管理 5. 統計的な分析とパーセンテージ計算

実装のポイント: - C90準拠のため、すべての変数を関数先頭で宣言 - time(NULL)を使った乱数の初期化 - 複雑な勝敗判定ロジック - ユーザー入力の検証 - 詳細な結果分析とフィードバック

じゃんけんの勝敗ルール: - グー(1) > チョキ(2) - チョキ(2) > パー(3)  
- パー(3) > グー(1) \*/```