東京ITスクール
TOKYO IT SCHOOL

C 言語 練習問題 初級編

練習問題の構成

この練習問題は、下記構成で作成しています。

<0 章～11 章>
インプレス社 スッキリわかる C 言語入門(第 2 版)(以降テキストと表記)の章構成に合わせて作成しています。
テキストの章番号と練習問題の章番号は一致していますので、練習問題を解く際に、該当する章を参照しながら、進めてください。

<12 章以降>
東京 IT スクールのカリキュラムに応じた章立てにしています。
学習内容に合わせて、該当する章の練習問題を進めてください。

練習問題を行う環境

Visual Studio Community2022 を利用することを想定しています。

練習問題の解答例は、以下の環境でコンパイルし動作確認済みです。

環境:Visual Studio 2022 Developer Command Prompt v16.4.5
ソースコードの文字コード:UTF-8
コンパイル時のコマンドとオプション: cl /source-charset:utf-8

開発環境の設定により、コンパイルエラーやワーニングが発生する可能性もあることをご了承ください。

練習問題の種類について

<「試してみましょう！」という記載があるもの >
以下の点を目的とした練習問題です。

- テキストに載っている方法を少し応用してみること
- 実行結果から理解を深めること

<「試してみましょう！」という記載がないもの >
以下の点を目的とした練習問題です。

- テキストに載っていること、練習問題の解答を組み合せること
- 処理手順を自ら考えて作成すること

練習問題の注意事項

C 言語およびプログラミングの理解を深めることを目的として作成しています。
そのため、実際の現場では使わない記述方法や処理内容も含まれていますが、様々な記述方法を知ることで、より深い理解を得てください。

目次

0. プログラミングとは

Question00_01

試してみましょう！
C ドライブ配下に、新規フォルダと拡張子 C のファイルを作成し コマンドプロンプト※でコンパイルおよびプログラムの実行をしましょう。

< プログラムの条件 >

・ フォルダ名、ファイル名は以下の指示に従うこと
　フォルダ名:c_training_q00
　ファイル名:Question00_01.c

・ 下記の実行結果を表示するコードを記述すること
　実行結果

```
こんにちは
```

※Visual Studio 2022 Developer Command Prompt を利用しましょう。

Question00_02

試してみましょう！
Visual Studio Community 2022 で、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q00
  プロジェクト名:Question00_02
  ファイル名:Question00_02.c

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
こんばんは。
```

※以降の練習問題は、Visual Studio Community2022 を利用します。

1. C 言語プログラミングの基礎

Question01_01

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_01
  ファイル名:Question01_01.c

- printf()を利用すること

- 実行結果は 2 行で表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
出力の練習
1 回目
```

Question01_02

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_02
  ファイル名:Question01_02.c

- printf()を 2 回利用すること

- 実行結果は 1 行で表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
出力の練習 2 回目
```

Question01_03

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_03
  ファイル名:Question01_03.c

- printf()を 3 回利用すること

- 実行結果は 2 行で表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
出力の練習 3 回目
出力を終了します
```

Question01_04

試してみましょう！
下記、プログラムはコンパイルエラーが発生しています。
原因を特定し、正しく動作するようにプログラムを修正しましょう。

```
#include <stdio.h>

int main(void)

	printf("ようこそ C へ");

	return 0;
}
```

修正後のプログラムは、下記の<プログラムの条件>に合わせて、コンパイルし、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_04
  ファイル名:Question01_04.c

- エラーを修正し、下記の実行結果を表示するようにコードを編集すること
  実行結果

```
ようこそC へ
```

Question01_05

試してみましょう！
下記、プログラムは正常に動作しているものの、インデントや改行がなく、読みにくいコードになっています。
正しくインデント、改行を挿入し、ソースコードを読みやすく修正しましょう。

```
#include <stdio.h>

int main(void) {
printf("赤パジャマ¥n"); printf("青パジャマ¥n"); printf("黄パジャマ¥n");return
0;
}
```

修正後のプログラムは、下記の<プログラムの条件>に合わせて、コンパイルし、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_05
  ファイル名:Question01_05.c

- ソースコードを見やすく修正し、下記の実行結果を表示するようにコードを編集すること
  実行結果

```
赤パジャマ
青パジャマ
黄パジャマ
```

Question01_06

試してみましょう！
下記、プログラムは正常に動作しているものの、コメントがありません。
それぞれの printf 文の前に 1 行コメントを記述しましょう。

```
#include <stdio.h>

int main(void) {

	printf("表示を　");
	printf("行います¥n");
	printf("終了します¥n");

	return 0;
}
```

修正後のプログラムは、下記の<プログラムの条件>に合わせて、コンパイルし、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_06
  ファイル名:Question01_06.c

- コメント内容は、処理内容にあったものを各自で考え、記述すること

- コメントを追加し、下記の実行結果を表示するようにコードを編集すること
  実行結果

```
表示を　行います
終了します
```

Question01_07

試してみましょう！
下記、プログラムは正常に動作しているものの、コメントがありません。
main の前に 複数行コメントを記述しましょう。

```
#include <stdio.h>

int main(void) {

	printf("メッセージを表示する　");
	printf("プログラムです¥n");
	printf("プログラムを終了します¥n");

	return 0;
}
```

コメント追記後のプログラムは、下記の<プログラムの条件>に合わせて、コンパイルし、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q01
  プロジェクト名:Question01_07
  ファイル名:Question01_07.c

- コメント内容は、プログラムの処理内容、作成日、作成者を記述すること

- コメントに記載するプログラムの処理内容は、処理内容にあったものを各自で考え、記述すること

- コメントを追加し、下記の実行結果を表示するようにコードを編集すること
  実行結果

```
メッセージを表示する　プログラムです
プログラムを終了します
```

2. 変数と型

Question02_01

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_01
  ファイル名:Question02_01.c

- printf()を 4 回使用すること

- 「桁」「回」は文字リテラル、「10」は整数リテラル、「30.8」は浮動小数点リテラルを使用すること

- 浮動小数点リテラルは、小数点第 1 位までを表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
10 桁
30.8 回
```

<ヒント>
- 小数点第 2 位までを表示する場合は、“%.2f”と指定する
- 表示内容を改行するには、“¥n”をつける必要がある

Question02_02

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_02
  ファイル名:Question02_02.c

- 型の違う変数を 2 つ宣言し、値を代入し出力すること

- 変数名は、自分で決定し適切な名前を使用すること

- printf()を 2 回使用すること

- 浮動小数点型は、小数点第 1 位までを表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
変数の値は 50
変数の値は 10.5
```

Question02_03

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_03
  ファイル名:Question02_03.c

- 数字を扱う変数を 2 つ、文字列を扱う変数を 1 つ宣言し、値を代入し出力すること

- 変数名は、自分で決定し適切な名前を使用すること

- 文字列を扱うために、main()の前に、以下の宣言を記載し、文字列は String 型として扱うこと

```
typedef char String[1024];
```

- printf()を 3 回使用すること

- 浮動小数点リテラルは、小数点第 2 位までを表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
12
20.58
こんにちは
```

<ヒント>
- 文字列を表示する場合に、“%s“と指定すると、文字列が表示される

Question02_04

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_04
  ファイル名:Question02_04.c

- 数字を扱う変数を 1 つ宣言し、初期値を代入し表示、値を変更して表示すること

- 変数名は、自分で決定し適切な名前を使用すること

- printf()を 2 回使用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
最初の値は、12
変更後の値は、20
```

Question02_05

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_05
  ファイル名:Question02_05.c

- 定数名は、自分で決定し適切な名前を使用すること

- 定数を 1 つ宣言し、値を代入すること

- printf()を 1 回使用し、定数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
円周率は、3.14
```

Question02_06

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_06
  ファイル名:Question02_06.c

- 文字型 1 つ、整数型 3 つ、浮動小数型 2 つとそれぞれ違う型の変数を宣言すること

- 変数名は、自分で決定し適切な名前を使用すること

- printf()を 6 回使用し、sizeof 演算子を用いて、変数のバイト数を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果　※環境によっては、サイズが異なる場合があります。

```
sizeof char:1バイト
sizeof short:2バイト
sizeof int:4バイト
sizeof long:4バイト
sizeof float:4バイト
sizeof double:8バイト
```

<ヒント>

- テキスト p.55 に、sizeof 演算子の例があります。

Question02_07

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q02
  プロジェクト名:Question02_07
  ファイル名:Question02_07.c

- char 型を 4 つ宣言し、以下のように初期化すること

```
char a = 0101;
char b = 0x42;
char c = 67;
char d = 'D';
```

- printf()を 2 回使用し、以下のように指定して表示すること

```
printf("%d %d %d %d ¥n", a, b, c, d);
printf("%c %c %c %c ¥n", a, b, c, d);
```

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
65 66 67 68
A B C D
```

- 実行後、Windows10 標準の電卓アプリを起動し、プログラマモードにして、A、B、C、D それぞれの文字コードを 2 進数、8 進数、10 進数、16 進数に変換した値を確認してみましょう

<ヒント>

- テキスト p.747 E.10 ASCII 文字コード
  A、B、C、D それぞれの文字コードを確認してみましょう
- Windows10 標準の電卓アプリ、プログラマモードでは、HEX：16 進数、
- DEC：10 進数、OCT：8 進数、BIN：2 進数で表示されます

3. 式と演算子

Question03_01

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_01
  ファイル名:Question03_01.c

- printf()を 1 回だけ使用すること

- エスケープシーケンスを使用すること

- 「少年老い易く」と「学成り難し」の間は、空白文字ではなく、タブ文字を利用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
少年老い易く　学成り難し
一寸の光陰軽んずべからず
```

<ヒント>

- テキスト p.745 エスケープシーケンスを参考にどのように記述すればよいか考えましょう。

Question03_02

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_02
  ファイル名:Question03_02.c

- 変数を利用せずに、printf()を 5 回使用し、加算(+)、減算(-)、乗算(*)、徐算(/)、剰余(%)の算術演算子を利用して計算結果を表示すること

- エスケープシーケンス、プレースホルダを使用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
15＋6 は 21 です
15－6 は 9 です
15×6 は 90 です
15÷6 は 2 です
15÷6 の余りは 3 です
```

Question03_03

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_03
  ファイル名:Question03_03.c

- 整数型の変数を num1、num2 を 2 つ宣言し、値を代入すること

- num1 は初期値:25、num2 は初期値:5 を代入すること

- printf()を 4 回使用し、整数型変数の四則演算結果を整数型で表示すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
たし算の結果は、30
引き算の結果は、20
かけ算の結果は、125
割り算の結果は、5
```

Question03_04

試してみましょう！
下記のプログラムを変更して、代入演算子を利用するコードに編集しましょう。

```
#include <stdio.h>

int main(void) {

	int num = 10;
	printf("%dに", num);
	num = num + 5;
	printf("5を加えた値は%dです¥n", num);
	num = num - 3;
	printf("加算結果から3を引いた値は%dです¥n", num);
	return 0;
}
```

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_04
  ファイル名:Question03_04.c

- 新たな変数は、追加しないこと

- 演算子は、代入演算子を使用すること

- printf()の内容は、書き換えないこと

- コード編集前と同じ、下記の実行結果を表示するコードを記述すること
  実行結果

```
10に5を加えた値は15です
加算結果から3を引いた値は12です
```

Question03_05

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_05
  ファイル名:Question03_05.c

- 整数型の変数 inum を宣言し、値を代入すること

- inum の値は、printf()内で変化させること

- printf()を 5 回使用し、演算子は必ず printf 内で使用すること

- 演算子は、インクリメント(++)、デクリメント(--)を使用すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
inum の値は 10 です
inum に 1 をたすと 11 です

表示後に inum の値(11)から 1 を引きます

inum の値は 10 です
inum から 1 を引くと 9 です
```

<ヒント>
- テキスト p.96~p.97
  コラム「インクリメント・デクリメント演算子は単独で使う」

Question03_06

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_06
  ファイル名:Question03_06.c

- 整数型の変数を inum1、inum2 を 2 つ宣言し、値を代入すること

- inum1 の初期値は 15、inum2 の初期値は 5 を設定すること

- inum2 の値を 1 回目と 2 回目、2 回目と 3 回目の結果表示間で変化させること

- 演算子は、除算(/)、剰余(%)、インクリメント(++)を使用すること

- printf()を 6 回使用すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
1 回目 割り算の結果は、3、余りは、0
2 回目 割り算の結果は、2、余りは、3
3 回目 割り算の結果は、2、余りは、1
```

Question03_07

半径と円周率から、円周の長さと円の面積を算出するプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_07
  ファイル名:Question03_07.c

- 半径 5.0cm の円周の長さと、円の面積を算出し、表示すること

- 円周率は、定数を宣言し、値は 3.14 を使用すること

- 円周の長さは、2×半径×円周率、円の面積は、半径×半径×円周率で求めること

- 半径の値、計算結果を入れる適切な型の変数を用意すること

- キャスト演算子を用いないこと

- 変数名は、自分で決定し適切な名前を使用すること

- printf()を 2 回使用すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
円周の長さ:31.4
円の面積:78.5
```

Question03_08

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_08
  ファイル名:Question03_08.c

- 以下、2 つの変数を用いること
  変数名: inum、 変数の型: int、 初期値: 0
  変数名: dnum、 変数の型:double、初期値:10.5

- inum に dnum の値を代入すること、その際にキャスト演算子を用いること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
dnum(10.5)を inum に代入すると 10 です。
```

Question03_09

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_09
  ファイル名:Question03_09.c

- 以下、2 つの変数を用いること
  変数名: inum、 変数の型: int、 初期値: 10
  変数名: dnum、 変数の型:double、初期値: 0.0

- dnum に inum の値を代入すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
inum(10)を dnum に代入すると 10.0 です。
```

Question03_10

円周率と円周の長さから、半径と円の面積を求めるプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_10
  ファイル名:Question03_10.c

- 円周の長さが 18.84 ㎝の円の半径と、円の面積を算出し、表示すること

- 円周率は、定数を宣言し、値は 3.14 を使用すること

- 円周の長さを求める公式(2×半径×円周率)から、半径を算出すること

- 円の面積は、半径×半径×円周率で求めること

- 円周の長さ、半径を入れる変数と、計算結果を入れる変数を用意すること

- 変数名は、自分で決定し適切な名前を使用すること

- 円の半径は整数、円の面積は小数点以下第 2 位まで表示すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
円の半径:3
円の面積:28.26
```

Question03_11

台形の面積を求めるプログラムを作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_11
  ファイル名:Question03_11.c

- 上辺の長さが 4 ㎝、下辺の長さが 9cm、高さが 5 ㎝の台形の面積を算出すること

- 台形の面積を求める公式は、(上辺+下辺)×高さ÷2 を使うこと

- 上辺、下辺、高さは、整数型の変数、台形の面積は、浮動小数点型の変数で扱うこと

- キャストを用いて算出し、小数点以下第 1 位まで表示すること

- printf()を 1 回使用すること

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
台形の面積（4cm+9cm）×5cm÷2=32.5
```

Question03_12

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_12
  ファイル名:Question03_12.c

- 文字型変数を宣言し、初期値に ‘A’を代入すること

- 演算子とプレースホルダを使用して、printf()を 1 回使用して下記の結果を表示すること
  実行結果

```
ABC
```

<ヒント>

- テキスト p.747 E.10 ASCII 文字コード
  A、B、C それぞれの文字コードは、1 つずつ値が違うことがわかります。その特徴と生かして、どのようにすればよいか考えてみましょう。

Question03_13

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_13
  ファイル名:Question03_13.c

- rand()を使用して、乱数を作ること

- 表示する数値は、1~6 の数値とする。

- 演算子とプレースホルダを使用して、printf()を 1 回使用して下記の実行結果を表示すること
  実行結果 (太字:実行するたびに数値が変化)

```
サイコロを振ります 。
サイコロの目は 6 でした。
```

<ヒント>

- テキスト p.115~p.117 乱数生成の準備する命令と、乱数を作る命令を使います。

Question03_14

値引き率がランダムに決定し、1000 円の商品を購入した場合の値引き後の額、消費税額を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_14
  ファイル名:Question03_14.c

- rand()を使用して、乱数を作ること

- 値引き率は、5%~49%の数値とする。

- 値引き後の金額と消費税額を算出して表示すること

- 商品価格の算出に利用する値引き率は、浮動小数点型の変数を利用すること

- 消費税額は、浮動小数点型の変数を利用して算出し、整数で表示すること

- 消費税率は、浮動小数点型の定数として宣言し、税率 10%とすること

- 円マーク(¥)とパーセント(%)は、半角文字で表示すること

- エスケープシーケンスとプレースホルダを使用して、下記の結果を表示すること
  実行結果 (太字:実行するたびに変化)

```
本日の値引き率が決定しました！
あなたの値引き率は 7%です。
1000 円の商品の場合、税抜き¥930 で購入できます。
消費税額(1 円以下切り捨て)は、¥93 です。
```

<ヒント>
- パーセント(%)を表示するためには、“%%”とすると表示できます。
- 円マークは、テキスト p.745 エスケープシーケンスを参照しましょう。

Question03_15

Question03_14 のプログラムを改良して、プログラム実行時に商品の値段を入力できるようにしましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_15
  ファイル名:Question03_15.c

- rand()を使用して、乱数を作ること
- 値引き率は、5%~49%の数値とする。
- 値引き後の金額と消費税額を算出して表示すること
- 入力する商品の価格は、文字列で受け取り、整数に変換すること
- 商品価格の算出に利用する値引き率は、浮動小数点型の変数を利用すること
- 消費税額は、浮動小数点型の変数を利用して算出し、整数で表示すること
- 消費税率は、浮動小数点型の定数として宣言し、税率 10%とすること
- 円マーク(¥)とパーセント(%)は、半角文字で表示すること
- エスケープシーケンスとプレースホルダを使用して、下記の結果を表示すること
  実行結果 (赤字:入力値、太字:実行するたびに数値が変化)

```
商品価格を入力してください。:3500

商品の価格は、¥3500 です。
本日、あなたの値引き率は 16%です。
¥3500 の商品が、税抜き¥2940 で購入できます。
消費税額(1 円以下切り捨て)は、¥294 です。
値引き後の商品価格(税込み)は、¥3234 です。
```

<ヒント>
- テキスト p.117~p.119 を参考に、入力値を受け取り、整数に変換しましょう。

Question03_16

身長と体重を入力して、BMI と適正体重を算出するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_16
  ファイル名:Question03_16.c

- BMI と適正体重は、下記の算出式で求めること
  - BMI の算出式:体重(kg)÷身長(m)^2
  - 適正体重の算出式:身長(m)^2×22
- 身長は、cm 単位の整数での入力を促し、文字列で受け取り、整数に変換すること
- 体重は、kg 単位の整数での入力を促し、文字列で受け取り、整数に変換すること
- BMI 値、適正体重は、小数点第 1 位まで表示すること
- 必要な変数は、適切な型と名前を自分で決定し、宣言すること
- エスケープシーケンスとプレースホルダを使用して、下記の結果を表示すること
  実行結果 (赤字:入力値 太字:入力値により変化)

```
身長(cm)を整数値で入力してください。:167
体重(kg)を整数値で入力してください。:51
身長:167cm 体重: 51kg
BMI:18.3
適正体重:61.4kg
```

<ヒント>

- 身長は、cm→m への単位換算が必要
- 単位換算後の値を、浮動小数点型変数で扱うと算出しやすくなる

Question03_17

摂氏温度(°C)を入力すると、華氏温度(°F)を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_17
  ファイル名:Question03_17.c

- 摂氏温度(°C)→華氏温度の換算は、下記の算出式で求めること
  - 華氏温度=(9÷5)×摂氏温度+32

- 摂氏温度は、文字列で受け取り、整数値に変換すること

- 必要な変数は、適切な型と名前を自分で決定し、宣言すること

- エスケープシーケンスとプレースホルダを使用して、下記の結果を表示すること
  実行結果 (赤字:入力値 太字:入力値により変化)

```
温度(摂氏°C)を入力してください。: 21
摂氏温度 : 21.0° C 華氏温度: 69.8° F
```

Question03_18

華氏温度(°F)を入力すると、摂氏温度(°C)を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q03
  プロジェクト名:Question03_18
  ファイル名:Question03_18.c

- 華氏温度→摂氏温度(°C)の換算は、下記の算出式で求めること
  - 摂氏温度=(5÷9)×(華氏温度-32)

- 華氏温度は、文字列で受け取り、整数値に変換すること

- 必要な変数は、適切な型と名前を自分で決定し、宣言すること

- エスケープシーケンスとプレースホルダを使用して、下記の結果を表示すること
  実行結果 (赤字:入力値 太字:入力値により変化)

```
温度(華氏°F)を入力してください。: 70
華氏温度: 70.0° F 摂氏温度 : 21.1° C
```

4. 条件分岐と繰り返し

Question04_01

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_01
  ファイル名:Question04_01.c

- 以下の処理の流れで作成すること
  「処理を開始します。」と表示
  変数 iNumber を宣言し、初期値として 10 を代入
  if 文で iNumber の値が 10 以上かを評価
  10 以上の場合、「iNumber の値は 10 以上です。」と表示
  10 未満の場合、何もせず、次の処理へ
  「処理を終了します。」と表示
- if 文を 1 回使うこと
- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(iNumber が 10 の場合)

```
処理を開始します。
iNumber の値は 10 以上です。

処理を終了します。
```

- プログラムを iNumber の値を 9 に変更した場合に、実行結果が変わることを確認すること(プログラムを書き換えて、コンパイルし実行)
  実行結果(iNumber が 9 の場合)

```
処理を開始します。

処理を終了します。
```

Question04_02

試してみましょう！
Question04_01 を変更して、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_02
  ファイル名:Question04_02.c

- 以下の処理の流れで作成すること
  1. 「処理を開始します。」と表示

     変数 iNumber を宣言し、初期値として 10 を代入

     if 文で iNumber の値が 10 以上かを評価

     10 以上の場合、「iNumber の値は 10 以上です。」と表示

     10 未満の場合、「iNumber の値は 10 未満です。」と表示

     「処理を終了します。」と表示

- if-else 文を 1 回使うこと

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果(iNumber が 10 の場合)

```
処理を開始します。
iNumber の値は 10 以上です。

処理を終了します。
```

- プログラムを iNumber の値を 9 に変更した場合に、実行結果が変わることを確認すること(プログラムを書き換えて、コンパイルし実行)
  実行結果(iNumber が 9 の場合)

```
処理を開始します。
iNumber の値は 10 未満です。

処理を終了します。
```

Question04_03

if 文と if 文の間で変数の値を判断して、表示内容を変えるプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_03
  ファイル名:Question04_03.c

- 以下の処理の流れで作成すること
  1. 変数 iNumber を宣言し、初期値として 10 を代入
     「1 回目:」と表示
     if 文で iNumber の値が 10 以上かを評価
     10 以上の場合、「iNumber の値は 10 以上です。」と表示
     10 未満の場合、何もせず、次の処理へ
     iNumber をデクリメント
     「iNumber をデクリメントしました。」と表示
     「2 回目:」と表示
     if 文で iNumber の値が 10 以上かを評価
     10 以上の場合、「iNumber の値は 10 以上です。」と表示
     10 未満の場合、何もせず、次の処理へ
     「処理を終了します。」と表示

- if 文を 2 回使うこと

- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
1 回目:iNumber の値は 10 以上です
iNumber をデクリメントしました
2 回目:
処理を終了します
```

Question04_04

Question04_03 のプログラムを変更して、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_04
  ファイル名:Question04_04.c

- 以下の処理の流れで作成すること
  1. 変数 iNumber を宣言し、初期値として 10 を代入
     「1 回目:」と表示
     if 文で iNumber の値が 10 以上かを評価
     10 以上の場合、「iNumber の値は 10 以上です。」と表示
     10 未満の場合、「iNumber の値は 10 未満です。」と表示
     iNumber をデクリメント
     「iNumber をデクリメントしました。」と表示
     「2 回目:」と表示
     if 文で iNumber の値が 10 以上かを評価
     10 以上の場合、「iNumber の値は 10 以上です。」と表示
     10 未満の場合、「iNumber の値は 10 未満です。」と表示
     「処理を終了します。」と表示

- if-else 文を 2 回利用すること

- エスケープシーケンス、プレースホルダを使用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
1 回目:iNumber の値は 10 以上です。
number をデクリメントしました。
2 回目:iNumber の値は 10 未満です。

処理を終了します。
```

Question04_05

試してみましょう！
テストの点を格納する変数の値により表示結果を変えるプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_05
  ファイル名:Question04_05.c
- int 型の変数 point を宣言し、80 を代入すること
- 点数によって表示結果を変えること
  - 80 点以上の場合:「優秀！！」と表示
  - 50 点以上 79 点以下の場合:「平均的」と表示
  - 30 点以上 49 点以下の場合:「ギリギリ合格」と表示
  - それ以外の場合:「追試が必要」と表示
- テスト結果を表示後、「お疲れ様でした」と表示
- if-else if-else 文を利用すること
- エスケープシーケンス、プレースホルダを使用すること
- 下記の実行結果を表示するコードを記述すること

実行結果 1:

```
テスト結果:80 点
評価:優秀！！
お疲れ様でした
```

実行結果 2:

```
テスト結果:50 点
評価:平均的
お疲れ様でした
```

実行結果 3:

```
テスト結果:30 点
評価:ギリギリ合格
お疲れ様でした
```

実行結果 4:

```
テスト結果:29 点
評価:追試が必要
お疲れ様でした
```

- 各条件に当てはまるように変数 point の値を書き換えて、コンパイル、実行し動作を確認すること
  - point の値が 80 の場合:実行結果 1
  - point の値が 50 の場合:実行結果 2
  - point の値が 30 の場合:実行結果 3
  - point の値が 29 の場合:実行結果 4

Question04_06

Question04_05 を変更して、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_06
  ファイル名:Question04_06.c

- 0~100 の乱数を発生させて、テスト結果の点数として表示すること
- 点数によって表示結果を変えること
  - 80 点以上の場合:「優秀！！」と表示
  - 50 点以上 79 点以下の場合:「平均的」と表示
  - 30 点以上 49 点以下の場合:「ギリギリ合格」と表示
  - それ以外の場合:「追試が必要」と表示
- テスト結果を表示後、「お疲れ様でした」と表示
- if-else if-else 文を利用すること
- エスケープシーケンス、プレースホルダを使用すること
- 下記の実行結果を表示するコードを記述すること

実行結果 1:

```
テスト結果:86 点
評価:優秀！！
お疲れ様でした
```

実行結果 2:

```
テスト結果:52 点
評価:平均的
お疲れ様でした
```

実行結果 3:

```
テスト結果:34 点
評価:ギリギリ合格
お疲れ様でした
```

実行結果 4:

```
テスト結果:23 点
評価:追試が必要
お疲れ様でした
```

Question04_07

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。
<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_07
  ファイル名:Question04_07.c
- 以下の処理の流れで作成すること
  1. 「処理を開始します。」と表示
     bool 型変数 flg を宣言し、初期値として false を代入
     if 文の条件式に「flg」と書き、true か false かを評価
       true の場合、「1 回目:flg の値は true です。」と表示
       false の場合、「1 回目:flg の値は false です。」と表示
     if 文の条件式に「!flg」と書き、true か false かを評価
       true の場合、「2 回目:!flg の値は true です。」と表示
       false の場合、「2 回目:!flg の値は false です。」と表示
     「処理を終了します。」と表示
- if-else 文を 2 回使うこと
- エスケープシーケンス、プレースホルダを使用して、変数の値を表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(flg が true の場合)

```
処理を開始します。
1 回目:flg の値は true です。
2 回目:!flg の値は false です。

処理を終了します。
```

- 変数 flg の初期値を true に変更した場合、実行結果が変わることを確認すること
  (プログラムを書き換えて、コンパイルし実行)
  実行結果(flg が false の場合)

```
処理を開始します。
1 回目:flg の値は false です。
2 回目:!flg の値は true です。

処理を終了します。
```

<ヒント>

- テキスト p.137~p.140 の bool 型の扱いを参照して、どうするか考えましょう

Question04_08

乱数生成した数値が 50 以上なら true、50 未満なら false を bool 型変数に代入し、結果を表示する処理を下記<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_08
  ファイル名:Question04_08.c

- 0~100 の乱数を発生させて、50 以上か、50 未満かを判断すること
- 判断した結果は、bool 型変数に代入すること
  50 以上の場合:true
  50 未満の場合:false
- bool 型変数を if-else 文の条件式で評価することその際に比較演算子ではなく、否定演算子を使用すること
- プログラム全体で、if-else 文を 2 回使用すること
- 値と判断結果を表示すること
- エスケープシーケンス、プレースホルダを使用すること
- 下記の実行結果になるように、タブ文字を使用すること

実行結果 1:

```
値は 47　判断結果は、50 未満でした
```

実行結果 2:

```
値は 67　判断結果は、50 以上でした
```

<ヒント>

- テキスト p.137~p.140 の bool 型の扱いを参照して、どうするか考えましょう

Question04_09

下記のコードは、値が 2 の場合に「値は 2 です」と表示することを想定していますが、想定通りに動きません。
間違っている箇所を見つけて、想定通りの結果になるようにコードを修正しましょう。

<想定している結果>
- 乱数で生成した値が、2 の場合、「値は 2 です」と表示
- 乱数で生成した値が、1 以上(2 は含まない)の場合、「値は 1 以上です」と表示
- 乱数で生成した値が、それ以外の場合、「値はそれ以外です」と表示

<修正前のコード>

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {

	srand(time(NULL));
	int iRandnum;
	//0～5 の乱数より-2 することで-2～3 の値にする
	iRandnum = rand() % 6-2;
//	----修正対象ここから----
	if (iRandnum >= 1) {
		printf("値は 1 以上です¥n");
	} else if (iRandnum == 2) {
		printf("値は 2 です¥n");
	} else {
		printf("値はそれ以外の数値です¥n");
	}
//	----修正対象ここまで----
	printf("¥n 値は %d でした。処理を終了します¥n",iRandnum);
	return 0;
}
```

修正する際に次ページの<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_09
  ファイル名:Question04_09.c

- 前ページの<修正前のコード>を入力して、動作確認をしたのち、修正箇所を検討すること
- 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること
- エスケープシーケンス、プレースホルダを使用すること
- 下記の実行結果になること

実行結果 1:

```
値は 2 です

値は 2 でした。処理を終了します
```

実行結果 2:

```
値は 1 以上です

値は 1 でした。処理を終了します
```

実行結果 3:

```
値はそれ以外の数値です

値は -2 でした。処理を終了します
```

Question04_10

下記のコードは、入力値が 2 回とも 1 の場合にメッセージを表示するプログラムです。if 文が多重になっている構造を、論理演算子を使って、1 つの if 文で判断できるようにコードを修正しましょう。

<修正前のコード>

```
#include <stdio.h>
#include <stdlib.h>

typedef char String[1024];
int main(void) {
	String strInput="";
	int inum1 = 0;
	int inum2 = 0;

	printf("1 か 2 を入力してください¥n");
	scanf("%s",strInput);
	inum1 = atoi(strInput);

	printf("もう一度 1 か 2 を入力してください¥n");
	scanf("%s",strInput);
	inum2 = atoi(strInput);

//	----修正対象ここから----
	if (inum1 == 1) {
		if (inum2 == 1) {
			printf("1 が 2 回入力されました¥n");
		}
	}
//	----修正対象ここまで----
	return 0;
}
```

修正する際に次ページの<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_10
  ファイル名:Question04_10.c

- <修正前のコード>を入力して、動作確認をしたのち、修正箇所を検討すること
- 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること
- エスケープシーケンス、プレースホルダを使用すること
- 修正前と同じ以下の実行結果になること(赤字は入力値)

実行結果 1:

```
1 か 2 を入力してください
1
もう一度 1 か 2 を入力してください
2
```

実行結果 2:

```
1 か 2 を入力してください
1
もう一度 1 か 2 を入力してください
1
1 が 2 回入力されました
```

Question04_11

試してみましょう！
下記のコードは、入力値が 1 か 2 の場合にメッセージを表示するプログラムです。
if-else 文を使用している構造のため、同じメッセージを表示する処理がプログラム中に 2 つあります。
論理演算子を使って、メッセージの表示を 1 つにできるようにコードを修正しましょう。

<修正前のコード>

```
#include <stdio.h>
#include <stdlib.h>

typedef char String[1024];
int main(void) {
	String strInput="";
	int inum = 0;

	printf("1 桁の数値を入力してください¥n");
	scanf("%s",strInput);
	inum = atoi(strInput);

//  -----修正対象ここから-----
	if (inum == 1) {
		printf("1 か 2 が入力されました¥n");
	}else if (inum == 2) {
		printf("1 か 2 が入力されました¥n");
	}else {
		printf("1 と 2 以外が入力されました¥n");
	}

//  -----修正対象ここまで-----
	return 0;
}
```

修正する際に次ページの<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_11
  ファイル名:Question04_11.c

- <修正前のコード>を入力して、動作確認をしたのち、修正箇所を検討すること

- 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること

- 修正前と同じ以下の実行結果になること(赤字:入力値)

実行結果 1:

```
1 桁の数値を入力してください
1
1 か 2 が入力されました
```

実行結果 2:

```
1 桁の数値を入力してください
2
1 か 2 が入力されました
```

実行結果 3:

```
1 桁の数値を入力してください
3
1 と 2 以外が入力されました
```

Question04_12

while 文を利用して 1 から 10 の値を当てるプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_12
  ファイル名:Question04_12.c
- 以下の処理の流れで作成すること
  1. 必要な変数を宣言し初期化
  2. 1～10 の乱数を生成
  3. while 文で以下の処理を繰り返す
     チャレンジ回数の表示
     「1～10 の数値を入力してください」と表示
     入力値を受け取る
  4. while 文終了後、「あたり！」と乱数の値を表示
  5. チャレンジ回数を表示
- while 文の繰り返し条件は、比較演算子を使うこと
- 下記の実行結果になるように、コードを記述すること
  実行結果(赤字:入力値、太字:実行時に変化)

```
1 回目のチャレンジ！
1～10 の数値を入力してください
10
2 回目のチャレンジ！
1～10 の数値を入力してください
9
3 回目のチャレンジ！
1～10 の数値を入力してください
8
4 回目のチャレンジ！
1～10 の数値を入力してください
7
あたり！！:値は 7
チャレンジ回数は、4 でした
```

Question04_13

while 文と if-else 文を利用して 5 以上の値が入力されるまで、入力を促すプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q04
  プロジェクト名:Question04_13
  ファイル名:Question04_13.c

- while 文の繰り返し条件は、bool 型変数と否定演算子を使うこと

- bool 型変数の初期値は、false にすること

- 5 未満が入力されたとき、bool 型変数に false を代入、5 以上が入力されたとき、bool 型変数に true を代入すること

- 値の入力処理は、while 文の中で記述すること

- 下記の実行結果になるように、メッセージを表示するコードを記述すること
  実行結果(赤字:入力値)

```
5 以上の数値を入力してください
4
5 以上の数値を入力してください
3
5 以上の数値を入力してください
2
5 以上の数値を入力してください
7
5 以上の数値が入力されました
処理を終了します
```

5. 制御構文のバリエーション

Question05_01

下記のコードは、月の日数が何日あるかを表示するプログラムです。
if 文を使っているため、少し読みにくいコードになっています。switch 文に置き換えて、コードの読みやすさを向上させましょう。

<修正前のコード>

```
#include <stdio.h>
#include <stdlib.h>

typedef char String[1024];

int main(void) {

	String strInput="";
	int inum = 0;

	printf("入力された月の日数を表示します。¥n");
	printf("1～12 の数値を入力してください。¥n");
	scanf("%s",strInput);
	inum = atoi(strInput);

//  ----修正対象ここから----
	if (inum == 2) {
		printf("2 月は 28 日もしくは 29 日です。");
	}else if (inum == 4 || inum == 6 || inum == 9 || inum == 11) {
		printf("%d 月は 30 日です。",inum);
	}else if (inum == 1 || inum == 3 || inum == 5 || inum == 7
		|| inum == 8 || inum == 10 || inum == 12) {
		printf("%d 月は 31 日です。",inum);
	}else {
		printf("%d 月はありません。",inum);
	}
//  ----修正対象ここまで----
	return 0;
}
```

修正する際に次ページの<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

・ ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
　ソリューション名:c_training_q05
　プロジェクト名:Question05_01
　ファイル名:Question05_01.c

・ <修正前のコード>を入力して、動作確認をしたのち、修正箇所を検討すること

・ 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること

・ 修正前と同じ以下の実行結果になること(赤字:入力値)
　実行結果 1: 2 月の場合

```
入力された月の日数を表示します。
1～12 の数値を入力してください。
2
2 月は 28 日もしくは 29 日です。
```

　実行結果 2: 4 月、6 月、9 月、11 月の場合

```
入力された月の日数を表示します。
1～12 の数値を入力してください。
4
4 月は 30 日です。
```

　実行結果 3: 1 月、3 月、5 月、7 月、8 月、10 月、12 月の場合

```
入力された月の日数を表示します。
1～12 の数値を入力してください。
5
5 月は 31 日です。
```

　実行結果 4: 1～12 以外の数値が入力された場合

```
入力された月の日数を表示します。
1～12 の数値を入力してください。
13
13 月はありません。
```

Question05_02

do-while 文を利用して 1 から 10 の値を当てるプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。(処理内容は Question04_12 と同じです。)

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_02
  ファイル名:Question05_02.c
- 以下の処理の流れで作成すること
  1. 必要な変数を宣言し初期化

     1～10 の乱数を生成

     do-while 文で以下の処理を繰り返す

     チャレンジ回数の表示

     「1～10 の数値を入力してください」と表示

     入力値を受け取る

     do-while 文終了後、「あたり！」と乱数の値を表示

     チャレンジ回数を表示
- do-while 文の繰り返し条件は、比較演算子を使うこと
- 下記の実行結果になるように、コードを記述すること
  実行結果(赤字:入力値、太字:実行時に変化)

```
1 回目のチャレンジ！
1～10 の数値を入力してください
10
2 回目のチャレンジ！
1～10 の数値を入力してください
9
3 回目のチャレンジ！
1～10 の数値を入力してください
8
4 回目のチャレンジ！
1～10 の数値を入力してください
7
あたり！！:値は 7
チャレンジ回数は、4 でした
```

Question05_03

Question04_13 で作成したプログラムを、while 文で無限ループを作成し、if-else 文、break 文、continue 文を使うプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_03
  ファイル名:Question05_03.c

- while 文で無限ループを作ること

- 5 未満が入力されたとき、再度入力を促し、5 以上が入力されたとき、処理を終了すること

- 値の入力処理は、while 文の中で記述すること

- 下記の実行結果になるように、メッセージを表示するコードを記述すること
  実行結果(赤字:入力値)

```
5 以上の数値を入力してください
4
5 以上の数値を入力してください
3
5 以上の数値を入力してください
2
5 以上の数値を入力してください
7
5 以上の数値が入力されました
処理を終了します
```

Question05_04

試してみましょう！
下記のコードを 5 回目の処理で終了するように書き換えてください。ただし、for 文の条件は変更してはいけません。

<修正前のコード>

```
#include <stdio.h>

int main(void) {
      for (int i = 1; i <= 10; i++) {
//  ----修正対象ここから----
            printf("%d 回目の処理です¥n",i);
//  ----修正対象ここまで----
      }

      printf("処理を終了します");

      return 0;
}
```

修正する際に下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_04
  ファイル名:Question05_04.c

- 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること

- 下記の実行結果になること
  実行結果

```
1 回目の処理です
2 回目の処理です
3 回目の処理です
4 回目の処理です
5 回目の処理です
処理を終了します
```

Question05_05

下記のコードを偶数回のみメッセージを表示するように書き換えてください。ただし、for 文の条件は変更してはいけません。

<修正前のコード>

```
#include <stdio.h>

int main(void) {
      for (int i = 1; i <= 10; i++) {
//  ----修正対象ここから----
            printf("%d 回目の処理です¥n",i);
//  ----修正対象ここまで----
      }

      printf("処理を終了します");

      return 0;
}
```

修正する際に下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_05
  ファイル名:Question05_05.c

- 「//----修正対象ここから----」「//----修正対象ここまで----」のコメントに挟まれた部分のみ修正すること

- 下記の実行結果になること
  実行結果

```
2 回目の処理です
4 回目の処理です
6 回目の処理です
8 回目の処理です
10 回目の処理です
処理を終了します
```

Question05_06

試してみましょう！
入力された段の九九を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_06
  ファイル名:Question05_06.c

- 入力値は、文字列で受け取り、整数に変換すること

- for 文を 1 つ利用すること

- 下記の実行結果になること
  実行結果(赤字:入力値)

```
入力された段の九九を表示します
1～9 の数値を入力してください
5
5 の段   5   10   15   20   25   30   35   40   45
```

<ヒント>

- テキスト p.173 コード 5-7 九九の表を出力するを参考にどのように変更するかを考えましょう。

Question05_07

試してみましょう！
九九をすべて表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_07
  ファイル名:Question05_07.c

- for 文を 2 つ利用し、for 文をネストして利用すること

- 下記の実行結果になること
  実行結果

```
九九を表示します
1 の段   1   2   3   4   5   6   7   8   9
2 の段   2   4   6   8  10  12  14  16  18
3 の段   3   6   9  12  15  18  21  24  27
4 の段   4   8  12  16  20  24  28  32  36
5 の段   5  10  15  20  25  30  35  40  45
6 の段   6  12  18  24  30  36  42  48  54
7 の段   7  14  21  28  35  42  49  56  63
8 の段   8  16  24  32  40  48  56  64  72
9 の段   9  18  27  36  45  54  63  72  81
```

<ヒント>

- テキスト p.173 コード 5-7 を参考に、数字と数字の間にタブ文字、各段の段数を表示するにはどうすればよいか考えましょう。

Question05_08

for 文を利用して、図を描くプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_08
  ファイル名:Question05_08.c

- for 文を 2 つ利用し、for 文をネストして利用すること

- printf 文は、プログラム全体で 2 つ利用すること

- 下記の実行結果になること
  実行結果

```
*****
*****
*****
```

Question05_09

Question05_08 を改良して、縦の数、横の数を指定できるプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_09
  ファイル名:Question05_09.c

- for 文を 2 つ利用し、for 文をネストして利用すること

- 入力値は文字列で受け取り、整数に変換すること

- printf 文は、プログラム全体で 5 つ利用すること

- 下記の実行結果になること
  実行結果(赤字:入力値、太字:入力値により変化)

```
縦の数:1～9 の値を入力してください
4
横の数:1～9 の値を入力してください
9
縦:4　横:9 の四角形を表示します。
*********
*********
*********
*********
```

Question05_10

for 文を利用して、直角三角形を描くプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_10
  ファイル名:Question05_10.c

- for 文を 2 つ利用し、for 文をネストして利用すること

- printf 文は、プログラム全体で 2 つ利用すること

- 下記の実行結果になること
  実行結果

```
*
**
***
****
*****
```

Question05_11

for 文と if 文を利用して、三角形を描くプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_11
  ファイル名:Question05_11.c

- for 文を 3 つ利用し、for 文をネストして利用すること

- printf 文は、プログラム全体で 4 つ利用すること

- 下記の実行結果になること
  実行結果

```
     *
    @@@
   *****
  @@@@@@@
 *********
@@@@@@@@@@@
***********
@@@@@@@@@@@@@
```

Question05_12

for 文と if 文を利用して、X を描くプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_12
  ファイル名:Question05_12.c

- for 文を 2 つ利用し、for 文をネストして利用すること

- printf 文は、プログラム全体で 3 つ利用すること

- 下記の実行結果になること
  実行結果

```
*     *
 *   *
  * *
   **
   **
  * *
 *   *
*     *
```

Question05_13

for 文を利用して、100 までの数字を表示するプログラムを作りましょう。
ただし、3 で割り切れる場合は「Fizz」、5 で割り切れる場合は「Buzz」、3 と 5 で割り切れる場合は「FizzBuzz」と表示します。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_13
  ファイル名:Question05_13.c
- for 文と if-else if-else 文を利用すること
- printf 文は、プログラム全体で 4 つ利用すること
- 下記の実行結果になること

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
～省略～

14
FizzBuzz
16

～省略～

89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

Question05_14

Question05_13 で for 文を利用して作成したプログラムを、while 文を利用するプログラムに変更しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_14
  ファイル名:Question05_14.c
- while 文と if-else if-else 文を利用すること
- while 文の条件式は、i<100 とすること
- printf 文は、プログラム全体で 4 つ利用すること
- 下記の実行結果になること(Question05_13 と同じ)

  実行結果

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
～省略～

14
FizzBuzz
16

～省略～

89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

Question05_15

Question05_13 で for 文を利用して作成したプログラムを、do-while 文を利用するプログラムに変更しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q05
  プロジェクト名:Question05_15
  ファイル名:Question05_15.c
- do-while 文と if-else if-else 文を利用すること
- do-while 文の条件式は、i<=100 を指定すること
- printf 文は、プログラム全体で 4 つ利用すること
- 下記の実行結果になること(Question05_13、Question05_14 と同じ)
  実行結果

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
～省略～

14
FizzBuzz
16

～省略～

89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

6. 構造体

Question06_01

試してみましょう！
猫の名前と年齢を管理するための構造体を作成し、値を代入、表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_01
  ファイル名:Question06_01.c

- 名前は文字列、年齢は整数で管理できるように構造体を定義すること

- 構造体の定義は、typedef 宣言を同時に行うこと

- 構造体メンバの初期化は、宣言と同時に行うこと

- 下記の実行結果になること
  実行結果

```
名前はコタロウです
年齢は 7 歳です
```

Question06_02

試してみましょう！
Question06_01 で作成した構造体に下記のメンバを追加し、猫のプロフィールを表示するプログラムを作りましょう。

・身長　浮動小数点型
・体重　浮動小数点型
・好きな食べ物　文字列型

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_02
  ファイル名:Question06_02.c

- 名前は文字列、年齢は整数で管理できるように構造体を定義すること

- 構造体の定義は、typedef 宣言を同時に行うこと

- 構造体メンバの初期化は、宣言と同時に行うこと

- 下記の実行結果になること
  実行結果

```
名前はコタロウです
年齢は 7 歳です
身長は 52.3cm です
体重は 4.8kg です
好きな食べ物はささみです
```

Question06_03

試してみましょう！
ゲームのキャラクターを複数作成するための構造体を用意し、値を代入、表示するプログラムを作りましょう。キャラクターは、職業、技名、攻撃力が決まっています。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_03
  ファイル名:Question06_03.c

- 職業、技名は文字列、攻撃力は整数で管理できるように構造体を定義すること

- 構造体の定義は、typedef 宣言を同時に行うこと

- 構造体メンバの初期化は、宣言と同時に行うこと

- 下記の実行結果になること
  実行結果

```
キャラクター紹介
キャラ 1：勇者/得意技：スマッシュソード/攻撃力：450
キャラ 2：魔女/得意技：聖なる祈り/攻撃力：380
キャラ 3：スライム/得意技：溶ける/攻撃力：10
```

Question06_04

Question06_03 で作成した構造体に一部追記して、簡単なゲームプログラムを作りましょう。また、ユーザーが入力した値に応じて処理を分岐させ、実行結果が変わるようにしましょう。その他下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行してください。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_04
  ファイル名:Question06_04.c

- Question06_03 の構造体に、整数値の hp を追加する

- 構造体の定義は、typedef 宣言を同時に行うこと

- 構造体メンバの初期化は、宣言と同時に行うこと

- 以下の実行結果になるコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
キャラクター紹介
キャラ 1：勇者/得意技：スマッシュソード/攻撃力：450/体力：300
キャラ 2：魔女/得意技：聖なる祈り/攻撃力：380/体力：300
キャラ 3：スライム/得意技：溶ける/攻撃力：10/体力：50

…
…
スライムが現れた！
戦うキャラクターを選択してください。1:勇者/2:魔女>1
勇者の攻撃！
スライムに、450 のダメージ
（スライムの HP が 0 になった場合）
スライムは倒された！
```

不正な値が入力された場合（入力値が 1、2 以外の場合）

```
（省略）
スライムが現れた！
戦うキャラクターを選択してください。1:勇者/2:魔女>3
不正な値が入力されました。ゲームを終了します。
```

Question06_05

下記のコードは、テストの合計点と平均点を表示するプログラムです。
科目ごとに異なる変数で扱っている状態のため、管理しにくい状態です。構造体を利用して、5 科目の点数と 5 科目の平均点を 1 つの塊として管理できるように変更しましょう。

<変更前のコード>

```
#include <stdio.h>

int main(void) {

	int iEnglish = 88;
	int iMath = 63;
	int iHistory = 54;
	int iScience = 76;
	int iGeography = 45;

	int iSum = iEnglish + iMath + iHistory + iScience + iGeography;

	printf("5 科目の合計点 : %d 点¥n", iSum);
	printf("5 科目の平均点 : %.1f 点¥n", (double)iSum/5);

	return 0;
}
```

下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_05
  ファイル名:Question06_05.c

- <変更前のコード>を入力して、動作確認をしたのち、変更箇所を検討すること
- 変更前と同じ以下の実行結果になること

  実行結果

```
5 科目の合計点:326
5 科目の平均点:65.2
```

Question06_06

Question06_05 で作成した構造体を利用して、5 科目の点数を入力し、平均点と合計点を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_06
  ファイル名:Question06_06.c

- 5 科目(英語、数学、歴史、科学、地理)の点数を順に入力してもらい、文字列型で受け取り、整数値に変換すること

- 構造体のメンバは、0 で初期化すること

- 以下の実行結果になるコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
英語の点数を入力してください:76

数学の点数を入力してください:45

歴史の点数を入力してください:89

科学の点数を入力してください:32

地理の点数を入力してください:99

5 科目の合計点:341
5 科目の平均点:68.2
```

Question06_07

Question06_06 で作成したコードを利用して、5 科目の平均点から判断して評価を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q06
  プロジェクト名:Question06_07
  ファイル名:Question06_07.c

- 5 科目(英語、数学、歴史、科学、地理)の点数を順に入力してもらい、文字列型で受け取り、整数値に変換すること

- 構造体のメンバは、0 で初期化すること

- 平均点の値によって、下記の評価を表示すること
  - 80 点以上の場合:「優秀！！」と表示
  - 50 点以上 79 点以下の場合:「平均的」と表示
  - 30 点以上 49 点以下の場合:「及第」と表示
  - それ以外の場合:「追試が必要」と表示

- 以下の実行結果になるコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
英語の点数を入力してください:76

数学の点数を入力してください:45

歴史の点数を入力してください:89

科学の点数を入力してください:32

地理の点数を入力してください:99

5 科目の合計点:341
5 科目の平均点:68.2
総合評価:平均的
```

7. 配列

Question07_01

試してみましょう！
下記のコードは、テストの合計点と平均点を表示するプログラムです。
科目ごとに異なる変数で扱っている状態のため、合計点の算出式が長くなってしまっています。配列を利用して 5 科目の点数を扱い、for 文を利用して 5 科目の合計点を算出できるように変更しましょう。

<修正前のコード>

```
#include <stdio.h>

int main(void) {

	int iEnglish = 88;
	int iMath = 63;
	int iHistory = 54;
	int iScience = 76;
	int iGeography = 45;

	int iSum = iEnglish + iMath + iHistory + iScience + iGeography;

	printf("5 科目の合計点 : %d 点¥n", iSum);
	printf("5 科目の平均点 : %.1f 点¥n", (double)iSum/5);

	return 0;
}
```

下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_01
  ファイル名:Question07_01.c

- <変更前のコード>を入力して、動作確認をしたのち、変更箇所を検討すること
- int 型の配列を宣言し、利用すること
- 変更前と同じ以下の実行結果になること

  実行結果

```
5 科目の合計点:326
5 科目の平均点:65.2
```

Question07_02

for 文を使って 9 個の値を持つ配列要素を下記のように入れ替えるプログラムを作成しましょう。

入れ替え前:10,20,30,40,50,60,70,80,90
入れ替え後:90,80,70,60,50,40,30,20,10

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_02
  ファイル名:Question07_02.c

- 9 個の整数値を格納する配列を使用すること

- 配列は、次の 9 個の値で初期化すること
- 10,20,30,40,50,60,70,80,90

- for 文を利用して、入れ替える前と、入れ替えた後を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
入れ替え前の状態: 10  20  30  40  50  60  70  80  90
入れ替え後の状態: 90  80  70  60  50  40  30  20  10
```

Question07_03

配列を使用して、10 個の数字を並べ替えるプログラムを作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_03
  ファイル名:Question07_03.c

- 10 個の整数値を格納する配列を使用すること

- 配列は、次の 10 個の値で初期化すること
- 31,41,59,26,53,58,97,93,23,84
- for 文を利用して、並べ替える前と、並べ替えた後を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
並べ替え前の状態: 31  41  59  26  53  58  97  93  23  84
並べ替え後の状態: 23  26  31  41  53  58  59  84  93  97
```

<ヒント>

- インターネットで、「アルゴリズム」、「ソート」というキーワードで検索し、並べ替えの手順を検討しましょう。

Question07_04

配列に格納した 10 個の数字の中に、入力値があるかどうかを調べるプログラムを作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_04
  ファイル名:Question07_04.c

- 10 個の整数値を格納する配列を使用すること

- 配列は、次の 10 個の値で初期化すること
- 31,41,59,26,53,58,97,93,23,84

- 入力値は、文字列型で受け取り、整数に変換すること

- 配列の中に、入力値があった場合、何番目に見つかったかを表示すること

- 下記の実行結果を表示するコードを記述すること
  (赤字:入力値、太字:入力値により変化)
  実行結果 1:見つかった場合

```
10～99 の 2 桁の数値を入力してください:84
84 は、9 の位置に見つかりました
```

  実行結果 2:見つからなかった場合

```
10～99 の 2 桁の数値を入力してください:22
22 は、見つかりませんでした
```

<ヒント>

- インターネットで、「アルゴリズム」、「探索」というキーワードで検索し、探索の手順を検討しましょう。

Question07_05

Question06_03 で作成した構造体に、苗字の頭文字を格納するメンバを追加し、誰のテスト結果表示かがわかるようにプログラムを変更しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_05
  ファイル名:Question07_05.c
- 苗字の頭文字を格納するメンバを char 型で追加すること
- アルファベット 1 文字で苗字の頭文字を入力してもらい、文字列型で受け取り、最初の 1 文字を char 型の構造体メンバに代入すること

- 5 科目(英語、数学、歴史、科学、地理)の点数を順に入力してもらい、文字列型で受け取り、整数値に変換すること

- 構造体のメンバは、0 で初期化すること

- 以下の実行結果になるコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
苗字の頭文字をアルファベット 1 文字で入力してください: A
英語の点数を入力してください : 12

数学の点数を入力してください : 34

歴史の点数を入力してください : 56

科学の点数を入力してください : 78

地理の点数を入力してください : 90

A さんのテスト結果

5 科目の合計点 : 270
5 科目の平均点 : 54.0
```

<ヒント>
- 文字列型は、typedef char String[1024];で定義していることから、char 型の配列であることがわかります。配列の 1 つ目の要素を取り出すにはどうすればよいか考えましょう

Question07_06

Question07_05 で変更した構造体を利用して、3 人のテスト結果を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_06
  ファイル名:Question07_06.c
- 構造体を格納する要素数 3 の配列を使用すること
- アルファベット 1 文字で苗字の頭文字を入力してもらい、文字列型で受け取り、最初の 1 文字を char 型の構造体メンバに代入すること
- 5 科目(英語、数学、歴史、科学、地理)の点数を順に入力してもらい、文字列型で受け取り、整数値に変換すること
- 構造体のメンバは、0 で初期化すること
- 以下の実行結果になるコードを記述すること
  実行結果(テストの点入力部分は省略、赤字:入力値、太字:入力値により変化)

```
---1 人目の情報を入力してください---
苗字の頭文字をアルファベット 1 文字で入力してください ： A
        ～ 省略 ～

---2 人目の情報を入力してください---
苗字の頭文字をアルファベット 1 文字で入力してください ： B
        ～ 省略 ～

---3 人目の情報を入力してください---
苗字の頭文字をアルファベット 1 文字で入力してください ： C
        ～ 省略 ～

---A さんのテスト結果---
5 科目の合計点 ： 313
5 科目の平均点 ： 62.6

---B さんのテスト結果---
5 科目の合計点 ： 356
5 科目の平均点 ： 71.2

---C さんのテスト結果---
5 科目の合計点 ： 187
5 科目の平均点 ： 37.4
```

Question07_07

科目名と点数を扱う構造体を定義し、5 科目の点数の高い順に並べ替えるプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_07
  ファイル名:Question07_07.c
- 科目と点数を扱う右のように構造体を定義し初期化すること
  - 文字列型　科目名
  - int 型　科目の点数

```
//科目名と点数の構造体
typedef struct{
  String strKamoku;
  int iScore;
}tagKamokuScore;

tagKamokuScore kamokuScore[]
      = {{"英語", 88},
         {"数学", 63},
         {"歴史", 54},
         {"科学", 76},
         {"地理", 45}};
```

- for 文を利用して、並べ替える前と、並べ替えた後を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
並べ替え前の状態：
        科目名:英語 点数:88
        科目名:数学 点数:63
        科目名:歴史 点数:54
        科目名:科学 点数:76
        科目名:地理 点数:45

並べ替え後の状態：
        科目名:英語 点数:88
        科目名:科学 点数:76
        科目名:数学 点数:63
        科目名:歴史 点数:54
        科目名:地理 点数:45
```

Question07_08

試してみましょう！
1 次元配列を 2 つ使った修正前のコードを、多次元配列を使ったプログラムに書き換えてみましょう。

<修正前のコード>

```
#include <stdio.h>

int main(void) {
	int num1[3];
	int num2[3];

	num1[0] = 10;
	num1[1] = 20;
	num1[2] = 30;
	num2[0] = 40;
	num2[1] = 50;
	num2[2] = 60;

	printf("1 段目の中身は%d です¥n", num1[0]);
	printf("2 段目の中身は%d です¥n", num1[1]);
	printf("3 段目の中身は%d です¥n", num1[2]);
	printf("4 段目の中身は%d です¥n", num2[0]);
	printf("5 段目の中身は%d です¥n", num2[1]);
	printf("6 段目の中身は%d です¥n", num2[2]);

	return 0;
}
```

下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。
<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_08
  ファイル名:Question07_08.c

- <変更前のコード>を入力して、動作確認をしたのち、変更箇所を検討すること
- int 型の多次元配列を宣言し、利用すること
- for 文を利用しないこと
- 変更前と同じ右の実行結果になること

```
1 段目の中身は 10 です
2 段目の中身は 20 です
3 段目の中身は 30 です
4 段目の中身は 40 です
5 段目の中身は 50 です
6 段目の中身は 60 です
```

Question07_09

試してみましょう！
Question07_08 で作成したプログラムを編集して、for 文を使ったプログラムに書き換えてみましょう。

下記の<プログラムの条件>に合うようにコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q07
  プロジェクト名:Question07_09
  ファイル名:Question07_09.c

- Question07_08.c をコピーして上記ファイル名で保存してから、編集すること

- for 文を 2 つ利用すること

- 変更前と同じ下記の実行結果になること

```
1 段目の中身は 10 です
2 段目の中身は 20 です
3 段目の中身は 30 です
4 段目の中身は 40 です
5 段目の中身は 50 です
6 段目の中身は 60 です
```

Question07_10

4 列 4 行の多次元配列を宣言し、列の内容と行の内容を入れ替えるプログラムを作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
・ ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
　ソリューション名:c_training_q07
　プロジェクト名:Question07_10
　ファイル名:Question07_10.c

・ int 型の多次元配列を宣言し、0~15 の初期値を代入すること

・ for 文を利用すること

・ 下記の実行結果になること
　実行結果

```
入れ替え前
   0    1    2    3
   4    5    6    7
   8    9   10   11
  12   13   14   15
入れ替え後
   0    4    8   12
   1    5    9   13
   2    6   10   14
   3    7   11   15
```

<ヒント>
・ 入れ替え用の多次元配列を宣言して、どうすればよいか考えてみましょう。

8. 関数

Question08_01

数値の比較をする関数を呼び出して、結果を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_01
  ファイル名:Question08_01.c

- 以下の関数を作ること
  - 関数名:mathMax
  - 引数:int 型 2 つ
  - 戻り値:int 型
  - 処理内容:2 つの引数を比較し、大きい値を戻り値として返す

- main 関数は、以下の処理を行うこと
  1. 2 つの値の入力を受け付ける
  2. 入力値を関数に渡す
  3. 関数の戻り値をメッセージに含めて表示する

- 入力値は、文字列で受け取り、整数型に変換すること

- 下記の実行結果を表示するコードを記述すること
  ※同値の場合の動作に関しては考慮しないものとする

実行結果(赤字:入力値、太字:入力値により変化)

```
2 桁の数値を 2 回入力してください
1 回目:54

2 回目:43
数値の比較をします
1 回目と 2 回目で大きい値は 54 です
```

Question08_02

Question08_01 では、2 つの入力値が同じ値の場合が判断できません。
数値の比較をする関数を修正して、1 つ目の引数が大きい場合、2 つの引数が同じ値、2 つ目の引数が大きい場合の 3 つの状態を戻り値として返すように変更しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_02
  ファイル名:Question08_02.c
- 以下の関数を作ること
  - 関数名:mathMaxCheck
  - 引数:int 型 2 つ
  - 戻り値:int 型
  - 処理内容:2 つの引数を比較し、以下の条件に基づいて値を返す
    1 つ目の引数が大きい場合:1
    2 つの引数が同じ値の場合:2
    2 つ目の引数が大きい場合:3

- 関数内で戻り値を格納する変数を宣言すること
- 関数内の return 文は 1 つだけにすること
- main 関数は、以下の処理を行うこと
  1. 2 つの値の入力を受け付ける

     入力値を関数に渡す

     関数の戻り値を switch 文の条件値として使用する

     switch 文内で状況に合わせたメッセージを表示する
- 入力値は、文字列で受け取り、整数型に変換すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

2 桁の数値を 2 回入力してください 1 回目:98 2 回目:87 数値の比較をします 1 回目と 2 回目で大きい値は 98 です	2 桁の数値を 2 回入力してください 1 回目:77 2 回目:77 数値の比較をします 1 回目と 2 回目の値は、77 で同じ値です

Question08_03

Question03_07 を参考に円周と円の面積を計算する関数を呼び出して、結果を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_03
  ファイル名:Question08_03.c
- 以下の関数を作ること
  - 関数名:calcCircleLength
  - 引数:double 型　半径
  - 戻り値:double 型　円周
  - 処理内容:引数から、円周を算出し戻り値として返す

  - 関数名:calcCircleArea
  - 引数:double 型　半径
  - 戻り値:double 型　円の面積
  - 処理内容:引数から、円の面積を算出し戻り値として返す

- 2 つの関数で共通して利用する円周率は、グローバル変数として宣言すること
- main 関数は、以下の処理を行うこと
  1. 半径の入力を受け付ける
  2. 入力値を関数に渡す
  3. 関数の戻り値をメッセージに含めて表示する
- 入力値は、文字列で受け取り、整数型に変換すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
半径を 1～9 の数値 1 桁で入力してください
半径:8
円周:50.24
円の面積:200.96
```

Question08_04

三角形の面積を計算する関数を呼び出して、結果を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_04
  ファイル名:Question08_04.c

- 以下の関数を作ること
  - 関数名:triangleCalcArea
  - 引数:int 型 2 つ
  - 戻り値:double 型
  - 処理内容:三角形の底辺と高さを引数として受け取り、面積を算出し戻り値として返す

- main 関数は、以下の処理を行うこと
  1. 底辺の長さ(int 型)、高さ(int 型)、面積(double 型)をメンバに持つ構造体を宣言し、0 で初期化する
     底辺の長さと高さの値の入力を受け付け、構造体メンバに格納する
     入力値を関数に渡す
     関数の戻り値を構造体メンバに格納する
     三角形の面積を表示する

- 入力値は、文字列で受け取り、整数型に変換すること

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
三角形の底辺の長さを 1～9 の 1 桁の数値で入力してください
底辺:5
三角形の高さを 1～9 の 1 桁の数値で入力してください
高さ:7
三角形の面積は 17.50 です
```

Question08_05

お弁当の消費税額を算出する関数を呼び出して、結果を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_05
  ファイル名:Question08_05.c

- 以下の関数を作ること
  - 関数名:tax8Calc
  - 引数:int 型　お弁当の本体価格(税抜き額)
  - 戻り値:int 型　消費税額
  - 処理内容:税率とお弁当の本体価格から、消費税額を算出し戻り値として返す

- main 関数は、以下の処理を行うこと
  1. お弁当の本体価格の入力を受け付ける
     入力値を関数に渡す
     関数の戻り値を使ってお弁当の税込額と消費税額を表示する

- 入力値は、文字列で受け取り、整数型に変換すること

- 消費税率は、グローバルスコープで定数として宣言すること

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
お弁当の本体価格を入力してください。
1000
税込み(8%)価格　：1080 円
消費税額(8%)　　：80 円
```

Question08_06

Question08_05 で作成したプログラムに、イートイン用の消費税額を算出する関数を追加し、持ち帰りとイートイン、それぞれのお弁当の税込額を表示しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_06
  ファイル名:Question08_06.c

- 以下の関数を追加すること
  - 関数名:tax10Calc
  - 引数:int 型　お弁当の本体価格(税抜き額)
  - 戻り値:int 型　消費税額
  - 処理内容:税率とお弁当の本体価格から、消費税額を算出し戻り値として返す

- main 関数は、以下の処理を行うこと
  1. お弁当の本体価格の入力を受け付ける
  2. 入力値を 8%用、10%用の関数に渡す
  3. それぞれの関数の戻り値を使ってお弁当の税込額と消費税額を表示する

- 入力値は、文字列で受け取り、整数型に変換すること

- 消費税率は、グローバルスコープで定数として宣言すること

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
お弁当の本体価格を入力してください。
1000
税込み(8%)価格　：1080 円
消費税額(8%)　　：80 円
税込み(10%)価格　：1100 円
消費税額(10%)　　：100 円
```

Question08_07

Question08_06 で作成したプログラムは、イートイン用の消費税額の算出関数、持ち帰り用消費税額の算出関数と類似の関数が 2 つある状態です。税額を引数として渡すことにより、消費税額が算出できる関数を 1 つに統合しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_07
  ファイル名:Question08_07.c

- tax8Calc と tax10Calc の 2 つの関数を以下の関数に変更すること
  - 関数名:taxCalc
  - 引数 1:int 型　お弁当の本体価格(税抜き額)
  - 引数 2:int 型　消費税率
  - 戻り値:int 型　消費税額
  - 処理内容:税率とお弁当の本体価格から、消費税額を算出し戻り値として返す

- main 関数は、以下の処理を行うこと
  1. お弁当の本体価格の入力を受け付ける
     入力値と消費税率を関数に渡す
     関数の戻り値を使ってお弁当の税込額と消費税額を表示する

- 入力値は、文字列で受け取り、整数型に変換すること

- 消費税率は、グローバルで定数として宣言すること

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
お弁当の本体価格を入力してください。
1000
税込み(8%)価格　：1080 円
消費税額(8%)　　：80 円
税込み(10%)価格 ：1100 円
消費税額(10%)　　：100 円
```

Question08_08

Question08_07 で作成したプログラムは、main 関数で消費税額の算出関数を 2 回呼び出し、税込額と消費税額の表示処理を 2 回ずつしています。表示用の関数を追加して重複している処理を減らしましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_08
  ファイル名:Question08_08.c
- 税込額と消費税額を表示する以下の関数を追加すること
  - 関数名:showPrice
  - 引数 1:int 型　お弁当の本体価格(税抜き額)
  - 引数 2:bool 型　イートイン:true　持ち帰り:false
  - 戻り値:なし
  - 処理内容:イートイン、持ち帰り種別とお弁当の本体価格から、消費税額を算出する関数を呼び出し、税込価格と消費税額を表示する
- main 関数は、以下の処理を行うこと
  2. お弁当の本体価格とイートインか持ち帰りかの入力を受け付ける

     入力値とイートインか持ち帰りかを関数に渡す

     関数の戻り値を使ってお弁当の税込額と消費税額を表示する
- 入力値は、文字列で受け取り、整数型に変換すること
- 消費税率は、グローバルで定数として宣言すること
- taxCalc 関数は、main 関数からではなく、showPrice 関数からのみ呼び出すこと
- 下記の実行結果を表示するコードを記述すること

```
お弁当の本体価格を入力してください。
1000
持ち帰りの場合:0、イートインの場合:1 を入力してください。
0
持ち帰りの税率:  8%
税込み(8%)価格   :  1080 円
消費税額(8%)     :  80 円
```

```
お弁当の本体価格を入力してください。
1000
持ち帰りの場合:0、イートインの場合:1 を入力してください。
1
イートインの税率:  10%
税込み(10%)価格  :  1100 円
消費税額(10%)    :  100 円
```

Question08_09

1-10 の乱数を取得する関数を呼び出して、占いプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q08
  プロジェクト名:Question08_09
  ファイル名:Question08_09.c

- 以下の関数を作ること
  - 関数名:uranai
  - 引数:無し
  - 戻り値:int 型
  - 処理内容:1-10 までの乱数を取得し、返す。

- main 関数は、以下の処理を行うこと
  1. 占いをするか、辞めるか、入力を受け付ける
  2. 入力値が 1 の場合は、関数(uranai)を呼び出し、処理が動く
  3. 関数の戻り値に応じてメッセージを変えて表示する

     1~3 の場合

     ```
     ラッキーナンバーは〇 （〇=1～3 の数値）
     新しい挑戦はなんでもうまくいく日！やったことのないことでも勇気を出してみて！
     ```

     4~6 の場合

     ```
     ラッキーナンバーは〇（〇=4～6 の数値）
     周りの人に助けられる一日かも？感謝を忘れずに言えると運気がさらにアップ！
     ```

     7~10 の場合

     ```
     ラッキーナンバーは〇（〇=7～10 の数値）
     難しい作業を任されて困ってしまう一日かも。できないことは周りに相談してみよう！
     ```

  4. 入力値が 2 の場合は、「システムを終了します。」とメッセージを表示する

- 入力値は、文字列で受け取り、整数型に変換すること
- 乱数生成には、stand,time,rand 関数を使用すること。
  参考 URL: https://monozukuri-c.com/langc-funclist-rand/

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
今日の運勢を占います！
1:ラッキーナンバー占い　2:今日は辞めておく　＞1

あなたの運勢を占います。
…
…
ラッキーナンバーは 2
新しい挑戦はなんでもうまくいく日！やったことのないことでも勇気を出してみて！

//2 を入力した場合の処理
今日の運勢を占います！
1:ラッキーナンバー占い　2:今日は辞めておく　＞2
システムを終了します。
```

Question08_10

1-6 の乱数を取得する関数を呼び出して、すごろくゲームプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

・ ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
ソリューション名:c_training_q08
プロジェクト名:Question08_10
ファイル名:Question08_10.c

・ 以下の関数を作ること
- 関数名:saikoro
- 引数:無し
- 戻り値:int 型
- 処理内容:1-6 までの乱数を取得し、返す

- 関数名:result
- 引数:int 型の変数
- 戻り値:void
- 処理内容:引数の値に応じて処理を変える
  ・引数の値が 1 の場合
  ① saikoro 関数を呼ぶ
  ② 「「〇マス」進みます。」とメッセージを表示。※〇＝乱数
  ・引数の値が 2 の場合
  ① 「パスしました。コンピューターのターンに移ります」とメッセージを表示する
  ② saikoro 関数を呼び出す。「「〇マス」進みました。」とメッセージを表示をする ※〇＝乱数

・ main 関数は、以下の処理を行うこと
1. さいころを回すか、パスをするか入力を受け付ける
2. 入力値が 1 の場合は、入力値を引数に渡したうえで
関数(result)を呼び出し、処理を実行する
3. すごろくを続けるかどうかメッセージを表示し、続ける場合は 1 の処理をもう一度繰り返す。不正な値が入力された場合は、「不正な値が入力されました。システムを終了します。」とメッセージを表示する

- 入力値は、文字列で受け取り、整数型に変換すること
- 乱数生成には、stand,time,rand 関数を使用すること。
  参考 URL: https://monozukuri-c.com/langc-funclist-rand/

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
すごろくゲームをスタートします！
あなたのターンです★★
1:さいころを回す　2:パス　＞1

5 マス進みます。

//2 を入力した場合の処理
1:さいころを回す　2:パス　＞2
パスしました。コンピューターのターンに移ります。
「6 マス」進みました。

すごろくを続けますか？
1:続ける　2:やめる＞1
→★★の箇所から同じ処理を再実行

//2 を入力した場合の処理
すごろくを続けますか？
1:続ける　2:やめる＞2
//そのまま終了

//0 以下または 3 以上を入力した場合の処理
すごろくを続けますか？
1:続ける　2:やめる＞3
不正な値が入力されました。システムを終了します。
```

9. アドレスとポインタ

Question09_01

試してみましょう！
ポインタの理解を深めるために、アドレス演算子(&)と間接演算子(*)を使ったプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_01
  ファイル名:Question09_01.c
- main 関数内で、右のように要素数 3 の int 型配列と、int 型のポインタ変数を 3 つ宣言し、初期化すること
- 間接演算子(*)を利用して、ポインタ型変数が指す値を表示すること
- ポインタ型変数の値（アドレス番地）を表示すること

```
//変数宣言と初期化
int inum[] = {1,2,3};

int *pnum1;
int *pnum2;
int *pnum3;

pnum1=&inum[0];
pnum2=&inum[1];
pnum3=&inum[2];
```

- アドレス番地の表示には、プレースホルダ“%p“を利用すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行時により変化)

```
*pnum1: 1       pnum1(inum[0]のアドレス番号): 008FF860
*pnum2: 2       pnum2(inum[1]のアドレス番号): 008FF864
*pnum3: 3       pnum3(inum[2]のアドレス番号): 008FF868
```

- 実行結果から、以下の点を確認すること
  - inum[2],inum[1],inum[0]のアドレス番地がそれぞれ異なっていること
  - inum[0]→inum[1]→inum[2]の順にアドレス番地が大きくなっていること

Question09_02

試してみましょう！
Question09_01 で int 型配列のアドレス番地を確認できました。char 型配列ではどのようにアドレス番地が変化するかを確認するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_02
  ファイル名:Question09_02.c
- char 型配列は、“ABCDE”で初期化すること
- char 型のポイント型変数は、配列として宣言すること
- 間接演算子(*)を利用して、ポインタ型変数が指す値を表示すること
- ポインタ型変数の値（アドレス番地）を表示すること
- アドレス番地の表示には、プレースホルダ“%p“を利用すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
*pmoji[0]: A    pmoji[0](cmoji1[0]のアドレス番地): 006FFC9C
*pmoji[1]: B    pmoji[1](cmoji1[1]のアドレス番地): 006FFC9D
*pmoji[2]: C    pmoji[2](cmoji1[2]のアドレス番地): 006FFC9E
*pmoji[3]: D    pmoji[3](cmoji1[3]のアドレス番地): 006FFC9F
*pmoji[4]: E    pmoji[4](cmoji1[4]のアドレス番地): 006FFCA0
```

- 実行結果から、以下の点を確認すること
  - char 型配列要素のアドレス番地が 1 バイトずつ変化していること
  - アドレス番地が配列の要素番号の順(0→1→2→3→4)に増えていること

Question09_03

試してみましょう！
ポインタの理解を深めるために、ポインタ型変数に間接演算子(*)をつけて、値を変更するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_03
  ファイル名:Question09_03.c
- main 関数内で、右のように、int 型変数、int 型のポインタ変数、char 型変数、char 型のポインタ変数をそれぞれ 1 つずつ宣言し、初期化すること
- 間接演算子(*)を利用して、ポインタ型変数が指す値を表示すること
- ポインタ型変数の値（アドレス番地）を表示すること
- アドレス番地の表示には、プレースホルダ“%p“を利用すること
- 以下の処理の流れで作成すること
  1. 初期値表示

     インクリメント演算子、間接演算子を使って値を変更

     - 変数の前にインクリメント演算子を置く前置インクリメントを使う
     - 書き方:++*pNum;　++*pMoji;

     前置インクリメントの結果を表示

     インクリメント演算子、間接演算子を使って値を変更

     - 丸かっこ()をつけて間接演算子の優先順位を高くする
     - 変数の後ろにインクリメント演算子を置く後置インクリメントを使う
     - 書き方: (*pNum)++;　(*pMoji)++;

     後置インクリメント()ありの結果を表示

     インクリメント演算子、間接演算子を使って値を変更

     - 変数の後ろにインクリメント演算子を置く後置インクリメントを使う
     - 書き方: *pNum++;　*pMoji++;

     後置インクリメント()なしの結果を表示
- 次ページの実行結果を表示するコードを記述すること

```
//変数宣言と初期化
int iNum =1;
int *pNum;
pNum = &iNum;

char cMoji = 'A';
char *pMoji;
pMoji = &cMoji;
```

実行結果(太字:環境、実行時により変化)

```
-----初期値を表示-----
*pNum : 1        pNum : 0057F964
*pMoji: A        pMoji: 0057F973
-----前置インクリメント[++*pNum, ++*pMoji]の結果を表示-----
*pNum : 2        pNum : 0057F964
*pMoji: B        pMoji: 0057F973
-----後置インクリメント()あり[(*pNum)++, (*pMoji)++]の結果を表示-----
*pNum : 3        pNum : 0057F964
*pMoji: C        pMoji: 0057F973
-----後置インクリメント()なし[*pNum++, *pMoji++]の結果を表示-----
*pNum : 5765492  pNum : 0057F968
*pMoji: シ       pMoji: 0057F974
```

- 実行結果から、以下の点を確認すること
  - インクリメント演算子をつける場所、()の有無で結果に違いがあること
  - ()なしで後置インクリメントの場合、アドレス番地が変化していること
  - ()なしで後置インクリメントの場合、値が全く違う値になっていること
  - *pMoji の値が、A→B→C と変化していること

<補足説明>
間接演算子(*)とインクリメント・デクリメント演算子は、評価の優先順位は同じです。
どちらの演算子も結合規則が右→左となっていることから、下記の評価順となり、結果に違いが出てしまいます。

- 前置インクリメントの場合　++*pNum
  評価順:変数 pNum と間接演算子を評価した後、インクリメント
  結果:pNum が指すアドレス番地に入っている値が変化
- 後置インクリメント()あり　(*pNum)++
  評価順:丸かっこ()内を評価した後、インクリメント
  結果:pNum が指すアドレス番地に入っている値が変化
- 後置インクリメント()なし　*pNum++
  評価順:変数 pNum をインクリメントした後、間接演算子を評価
  結果:pNum が指すアドレス番地の値が変化

<参考資料>

- テキスト p.89~p.91 演算子　優先順位の原則と結合規則の原則
- テキスト p.96~p.97 コラム「インクリメントとデクリメント演算子は単独で使う」
- テキスト p.734　E.3 演算子

Question09_04

試してみましょう！
Question09_03 でポインタ型変数に間接演算子(*)をつけて、値を変更することができ、文字に 1 加えていくと、A→B→C と変化することがわかりました。
文字型変数の値を代入で変化させるのではなく、ポインタ型変数が指す値を変化させることで、アルファベットの A から Z、Z から A を表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_04
  ファイル名:Question09_04.c

- main 関数内で、右のように char 型変数、char 型のポインタ変数をそれぞれ 1 つずつ宣言し、初期化すること

```
//変数宣言と初期化
char cMoji = 'A';
char *pMoji;
pMoji = &cMoji;
```

- 間接演算子(*)を利用して、ポインタ型変数が指す値を変化させること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
---A から Z を表示---
 <A> <B> <C> <D> <E> <F> <G> <H> <I> <J> <K> <L> <M>
 <N> <O> <P> <Q> <R> <S> <T> <U> <V> <W> <X> <Y> <Z>
---Z から A を表示---
 >Z< >Y< >X< >W< >V< >U< >T< >S< >R< >Q< >P< >O< >N<
 >M< >L< >K< >J< >I< >H< >G< >F< >E< >D< >C< >B< >A<
```

Question09_05

試してみましょう！
Question06_02 で作成したプログラムは、main 関数内で猫のプロフィールを表示しています。猫のプロフィールを表示する関数を追加し、関数を呼び出せば、プロフィールを表示できるようにしましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名 ：Question09_05
  ファイル名 ：Question09_05.c

- Question06_02.c をコピーして、上記ファイル名に変更すること

- 以下のプロフィール表示用関数を作ること
  - 関数名:showProfile
  - 引数:tagCat 構造体のアドレス
  - 戻り値:なし
  - 処理内容:構造体のメンバを参照し、値を表示する

- showProfile 関数内で構造体のメンバにアクセスする際に間接演算子(*演算子)を使うこと

- 下記の実行結果になること
  実行結果

```
名前はコタロウです
年齢は 7 歳です
身長は 52.3cm です
体重は 4.8kg です
好きな食べ物はささみです
```

<ヒント>
- テキスト p.350　ポインタ渡しを参考にどのように記述すればよいか考えましょう。

Question09_06

試してみましょう！
Question09_05 で作成したプログラムは、showProfile 関数内で間接演算子(*演算子)を利用して、構造体メンバにアクセスしました。この方法は、()をつけ忘れてしまうと正しい結果を得ることができません。
showProfile 関数内で間接演算子を利用している部分を、アロー演算子(->)に書き換えてみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_06
  ファイル名:Question09_06.c

- Question09_05.c をコピーして、上記ファイル名に変更すること

- showProfile 関数内のみ書き換えること

- showProfile 関数内で構造体のメンバにアクセスする際にアロー演算子を使うこと

- 下記の実行結果になること(Question09_05 と同じ)
  実行結果

```
名前はコタロウです
年齢は 7 歳です
身長は 52.3cm です
体重は 4.8kg です
好きな食べ物はささみです
```

<ヒント>
- テキスト p.351　コラム「アロー演算子」で使い方を確認しましょう

Question09_07

Question08_03 で作成した円周と円の面積を計算する関数を戻り値なしで計算結果を得る形に変更しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_07
  ファイル名:Question09_07.c
- Question08_03 で作成した関数を、以下に変更すること
  - 関数名:calcCircleLength
  - 引数:double 型　半径、double 型のアドレス　円周格納用のポインタ
  - 戻り値:なし
  - 処理内容:第 1 引数から、円周を算出し、円周格納用変数に値を入れる

  - 関数名:calcCircleArea
  - 引数:double 型　半径、double 型のアドレス　円の面積格納用のポインタ
  - 戻り値:なし
  - 処理内容:第 1 引数から、円の面積を算出し、円の面積格納用変数に入れる

- 2 つの関数で共通して利用する円周率は、グローバル変数として宣言すること
- main 関数は、以下の処理を行うこと
  1. 円周と円の面積を格納する変数を準備

     半径の入力を受け付ける

     入力値を関数に渡す

     円周算出関数を呼ぶ

     円の面積算出関数を呼ぶ

     円周と円の面積を表示する
- 入力値は、文字列で受け取り、整数型に変換すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
半径を 1～9 の数値 1 桁で入力してください
半径:8
円周:50.24
円の面積:200.96
```

Question09_08

Question08_08 で作成したプログラムは、表示用関数で、消費税額の計算処理関数を呼びだす構成になっています。お弁当の価格を扱う構造体を用意し、価格入力後に構造体メンバに値を代入する関数を作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_08
  ファイル名:Question09_08.c
- 以下の構造体を定義すること
  - int 型　本体価格
  - int 型　8%の消費税額
  - int 型　10%の消費税額
  - int 型　8%消費税額込みの価格
  - int 型　10%消費税額込みの価格

```
typedef struct {
        int price;
        int tax8Price;
        int tax10Price;
        int priceTax8in;
        int priceTax10in;
}tagLunchBox;
```

- 構造体を初期化する以下の関数を追加すること
  - 関数名: initLunchBoxData
  - 引数 1:int 型　お弁当の本体価格(税抜き額)
  - 引数 2:tagLunchBox 型　お弁当価格用構造体
  - 戻り値:なし
  - 処理内容:お弁当の本体価格から、消費税額、消費税込み価格を算出し、構造体メンバを初期化する。消費税額の算出は、tacCalc 関数を呼び出す
- お弁当価格を表示する showPrice 関数を以下のように変更すること
  - 関数名: showPrice
  - 引数 1: tagLunchBox 型　お弁当価格用構造体
  - 引数 2:bool 型 持ち帰りの場合 0、イートインの場合 1
  - 戻り値:なし
  - 処理内容:持ち帰りかイートインかを判断し、条件に合った値を構造体メンバから取り出し表示する

- main 関数は、以下の処理を行うこと
  1. お弁当の本体価格の入力を受け付ける

     構造体 tagLunchBox の変数を宣言する

     入力値と構造体のポインタを initLunchBoxPrice 関数に渡す

     持ち帰りかイートインかの入力を受け付ける

     構造体のポインタと持ち帰りかイートインかの入力値を showPrice 関数に渡す
- 本体価格の入力値は、文字列で受け取り、整数型に変換すること
- 持ち帰りか、イートインかの入力値は、文字列で受け取り、bool 型に変換すること
- 消費税率は、グローバルで定数として宣言すること
- taxCalc 関数は、main 関数からではなく、initLunchBoxPrice 関数からのみ呼び出すこと
- 下記の実行結果を表示するコードを記述すること(Question08_08 と同じ)

  実行結果(赤字:入力値、太字:入力値により変化)

```
お弁当の本体価格を入力してください。
1000
持ち帰りの場合:0、イートインの場合:1 を入力してください。
0
持ち帰りの税率: 8%
税込み(8%)価格　: 1080 円
消費税額(8%)　　: 80 円
```

```
お弁当の本体価格を入力してください。
1000
持ち帰りの場合:0、イートインの場合:1 を入力してください。
1
イートインの税率: 10%
税込み(10%)価格 : 1100 円
消費税額(10%)　 : 100 円
```

Question09_09

Question07_05 で作成したプログラムは、科目名をプログラム内に埋め込んでいるため、科目名が変更になった場合にプログラムを変更する必要があります。
科目数を 5 つに限定して、科目名と点数を扱う構造体、生徒の点数を扱う構造体を定義し、科目名を入力する処理、生徒の点数を入力する処理、平均点と合計点を計算する関数を用意しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_09
  ファイル名:Question09_09.c
- 科目と点数を扱う右のような構造体を定義すること
  - 文字列型　科目名
  - int 型　科目の点数

```
//科目名と点数の構造体
typedef struct{
  String strKamoku;
  int iScore;
}tagKamokuScore;
```

- 生徒の点数を扱う右のような構造体を定義すること
  - char 型 苗字の頭文字
  - int 型　科目数
  - 構造体型 科目名と点数
  - int 型　全科目の合計点
  - double 型　全科目の平均点

```
//生徒の点数を扱う構造体
typedef struct{
  char cInitmoji;
  int iKamokuCnt;
  tagKamokuScore scores[5];
  int iTotal;
  double dAvg;
}tagStudentScore;
```

- 科目名を入力する以下の関数を作成すること
  - 関数名: inputKamoku
  - 引数 1:tagStudentScore 型　生徒の点数を扱う構造体
  - 戻り値:なし
  - 処理内容:for 文を使い科目数回、科目名の入力を促し、入力値を構造体の科目名に格納する
- 生徒の点数を入力する以下の関数を作成すること
  - 関数名: inputScore
  - 引数 1:tagStudentScore 型　生徒の点数を扱う構造体
  - 戻り値:なし
  - 処理内容: for 文を使い科目数回、科目名ごとに点数の入力を促し、入力値を構造体の科目の点数に格納する

- 平均点と合計点を計算する以下の関数を作成すること
  - 関数名: calcTotalScore
  - 引数 1:tagStudentScore 型　生徒の点数を扱う構造体
  - 戻り値:なし
  - 処理内容: for 文を使い、全科目の合計点、全科目の平均点を算出し、構造体の合計点と平均点に格納する
- main 関数は、以下の処理を行うこと
  1. 生徒の点数を扱う構造体を用意し、科目数を代入する

     科目名を入力する処理 inputKamoku 関数を呼び出す

     苗字の頭文字の入力を促し、構造体のメンバに入力値を代入する

     科目ごとの点数を入力する処理 inputScore 関数を呼び出す

     全科目の合計点、平均点を求める処理 calcTotalScore 関数を呼び出す

     苗字の頭文字、全科目の合計点、平均点を表示する
- 科目名の入力値は文字列で受け取る
- アルファベット 1 文字で苗字の頭文字を入力してもらい、文字列型で受け取り、最初の 1 文字を char 型の構造体メンバに代入すること
- 科目ごとの点数の入力値は、文字列で受け取り、整数型に変換すること
- 下記の実行結果を表示するコードを記述すること

  実行結果(赤字:入力値、太字:入力値により変化)

```
1 つ目の科目名を入力してください: 国語
2 つ目の科目名を入力してください: 算数
3 つ目の科目名を入力してください: 理科
4 つ目の科目名を入力してください: 社会
5 つ目の科目名を入力してください: 英語
苗字の頭文字をアルファベット 1 文字で入力してください : A

A さんのテスト結果を入力してください
科目:国語の点数を入力してください : 66
科目:算数の点数を入力してください : 77
科目:理科の点数を入力してください : 88
科目:社会の点数を入力してください : 99
科目:英語の点数を入力してください : 11

---A さんのテスト結果---
5 科目の合計点 : 341
5 科目の平均点 : 68.2
```

Question09_10

4 列 4 行の多次元配列を宣言し、降順に並べ替えるプログラムを作成しましょう。
2 次元配列 array[][]は、「int *p; p=(int *)array;」とすることで、1 次元配列として扱ってみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q09
  プロジェクト名:Question09_10
  ファイル名:Question09_10.c

- int 型の多次元配列を宣言し、0~15 の初期値を代入すること

- for 文を利用すること

- 下記の実行結果になること
  実行結果

```
入れ替え前
   0    1    2    3
   4    5    6    7
   8    9   10   11
  12   13   14   15
入れ替え後
  15   14   13   12
  11   10    9    8
   7    6    5    4
   3    2    1    0
```

10. メモリアクセスのからくり

Question10_01

試してみましょう！
Question09_01 の実行結果を見ると、int 型配列 inum の各要素のアドレス番地が一定間隔で異なっていることが確認できます。

```
*pnum1: 1       pnum1(inum[0]のアドレス番号): 008FF860
*pnum2: 2       pnum2(inum[1]のアドレス番号): 008FF864
*pnum3: 3       pnum3(inum[2]のアドレス番号): 008FF868
```

ポインタ型変数の値を加減算した場合にどのような変化があるかを確認してみましょう。Question09_01.c にコードを追加して、ポインタ型変数 pnum2 の値を 1 つ増やした場合および、1 つ減らした場合の変化を確認するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_01
  ファイル名:Question10_01.c
- Question09_01.c をコピーして、Question10_01.c として編集すること
- (pnum2+1)のアドレス番地、pnum2 のアドレス番地、(pnum2-1)のアドレス番地を表示すること
- アドレス番地の表示には、プレースホルダ%p を利用すること
- 間接演算子(*)を利用して、ポインタ型変数が指す値を表示すること
- 添え字演算子([])を利用して、ポインタ型変数が指す値を表示すること

次ページに続く

- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
試してみよう！
pnum2+1 のアドレス番地 pnum2+1    = 00B3FB74
pnum2+1 の値 *(pnum2+1)           = 3
pnum2+1 の値 pnum2[0+1]           = 3

pnum2 のアドレス番地 pnum         = 00B3FB70
pnum2 の値 *(pnum2)               = 2
pnum2 の値 pnum2[0]               = 2

pnum2-1 のアドレス番地 pnum2-1    = 00B3FB6C
pnum2-1 の値 *(pnum2-1)           = 1
pnum2-1 の値 pnum2[0-1]           = 1
```

- 実行結果から、以下の点を確認すること
  - アドレス番地が一定間隔で変化していること
  - 間接演算子(*)と添え字演算子([])を使った場合で同じ値が表示されていること

Question10_02

試してみましょう！
Question09_02 で char 型配列のポイント型変数を配列で宣言し、配列要素に対応するアドレス番地を配列に格納する方法で扱いました。char 型配列の先頭要素を指すアドレス番地を格納するポインタ変数を用意し、ポインタ変数を扱って配列要素を表示するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_02
  ファイル名:Question10_02.c
- char 型配列は、“ABCDE”で初期化すること
- char 型のポイント型変数を 1 つ宣言し、char 型配列のアドレス番地を格納すること
- 間接演算子(*)を利用して、ポインタ型変数が指す値を表示すること
- ポインタ型変数の値（アドレス番地）を表示すること
- アドレス番地の表示には、プレースホルダ“%p“を利用すること
- for 文内で以下の処理を行い、配列要素の内容を表示すること
  1. 間接演算子を指定してポインタ型変数が指す値、配列要素、ポインタ型変数の値(アドレス番地)を表示
     ポインタ型変数をインクリメント
- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
*pmoji: A       cmoji[0]: A     pmoji(アドレス番号): 0055F830
*pmoji: B       cmoji[1]: B     pmoji(アドレス番号): 0055F831
*pmoji: C       cmoji[2]: C     pmoji(アドレス番号): 0055F832
*pmoji: D       cmoji[3]: D     pmoji(アドレス番号): 0055F833
*pmoji: E       cmoji[4]: E     pmoji(アドレス番号): 0055F834
```

- 実行結果から、以下の点を確認すること
  ➢ *pmoji の値と、cmoji[i]の値が同じであること
-

Question10_03

試してみましょう！
Question10_02 でポインタ変数を使って配列要素が表示できることがわかりました。
ポインタ変数を使って、配列要素の値を前から順、後から順に表示する処理を作成してみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_03
  ファイル名:Question10_03.c
- char 型配列は、“ABCDEFG”で初期化すること
- 配列の内容は、添え字を使わずに間接演算子(*)を利用して、値を表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果

```
前から順:A B C D E F G
後から順:G F E D C B A
```

Question10_04

試してみましょう！
int 型の配列を、memcpy 関数を使ってコピーしてみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_04
  ファイル名:Question10_04.c
- int 型の配列を 2 つ宣言し、下記のように 1 つは、1,2,3,4,5,6,7,8 で初期化すること

```
//変数宣言と初期化
int iArray[8] = {1,2,3,4,5,6,7,8};
int iArrayCopy[8];
```

- memcpy 関数を呼び出す前と呼び出した後の配列要素の値を表示すること
- memcpy で指定するコピーするバイト数は、sizeof 演算子を使って算出すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
---コピー前---
iArray[4]:        5
iArrayCopy[4]:    16617920
iArray アドレス番地:       012FFE7C
iArrayCopy アドレス番地:   012FFE5C
---コピー後---
iArray[4]:        5
iArrayCopy[4]:    5
iArray アドレス番地:       012FFE7C
iArrayCopy アドレス番地:   012FFE5C
```

- 実行結果から、以下の点を確認すること
  - 配列要素の値がコピーされていること
  - コピー前とコピー後で、コピー先の配列 iArrayCopy のアドレス番地が変化していないこと

Question10_05

試してみましょう！
構造体を、memcpy 関数を使ってコピーしてみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_05
  ファイル名:Question10_05.c
- 構造体を右のように宣言すること
- コピー元、、コピー先の構造体メンバに初期値を設定すること
- memcpy 関数を呼び出す前と呼び出した後の構造体メンバの値を表示すること
- memcpy で指定するコピーするバイト数は、sizeof 演算子を使って算出すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
//構造体の定義と宣言
typedef struct{
  int iSubNum;
}tagTestSubStruct;

typedef struct{
  char chMoji;
  int iNum;
  tagTestSubStruct subStruct;
}tagTestStruct;

tagTestStruct structOrg;
tagTestStruct structCpy;
```

```
---コピー前---
 <<<コピー元>>>
structOrg.chMoji                A
structOrg.iNum:                 0
structOrg.subStruct.iSubNum:    9
structOrg アドレス番地:   00E2FBC4
 <<<コピー先>>>
structCpy.chMoji                Z
structCpy.iNum:                 5
structCpy.subStruct.iSubNum:    4
structCpy アドレス番地:   00E2FBB8
---コピー後---
 <<<コピー元>>>
structOrg.chMoji                A
structOrg.iNum:                 0
structOrg.subStruct.iSubNum:    9
structOrg アドレス番地:   00E2FBC4
 <<<コピー先>>>
structCpy.chMoji                A
structCpy.iNum:                 0
structCpy.subStruct.iSubNum:    9
structCpy アドレス番地:   00E2FBB8
```

- 実行結果から、以下の点を確認すること
  - 構造体メンバの値がコピーされていること
  - コピー前とコピー後で、コピー先のアドレス番地が変化していないこと

Question10_06

試してみましょう！
int 型配列を、memcmp 関数を使って、比較してみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_06
  ファイル名:Question10_06.c
- int 型配列を右のように宣言し、初期化すること

```
int iArray1[5] = {12,34,56,78,90};
int iArray2[5] = {12,34,56,78,90};
int *pArray;
pArray = &iArray1[0];
```

- iArray1,iArray2,pArray のアドレス番地、要素の値を表示すること
- ==演算子で比較した場合(等値判定)と memcmp で比較した場合(等価判)の違いがわかるように下記の 6 パターンで比較すること
  - パターン 1：iArray1 と iArray2 を==演算子で比較
  - パターン 2: iArray1 と pArray を==演算子で比較
  - パターン 3: iArray2 と pArray を==演算子で比較
  - パターン 4: iArray1 と iArray2 を memcmp で比較
  - パターン 5: iArray1 と pArray を memcmp で比較
  - パターン 6: iArray2 と pArray を memcmp で比較

- memcmp で指定するバイト数は、sizeof 演算子を使って算出すること
- 次ページの実行結果を表示するコードを記述すること

実行結果(太字:環境、実行タイミングにより変化)

```
<<< iArray1、iArray2、pArray のアドレス番地を表示 >>>
iArray1=0113FB40 ,iArray2=0113FB2C ,pArray=0113FB40

<<< iArray1、iArray2、pArray の値を表示 >>>
iArray1[0]=12 ,iArray2[0]=12 ,*(pArray+0)=12
iArray1[1]=34 ,iArray2[1]=34 ,*(pArray+1)=34
iArray1[2]=56 ,iArray2[2]=56 ,*(pArray+2)=56
iArray1[3]=78 ,iArray2[3]=78 ,*(pArray+3)=78
iArray1[4]=90 ,iArray2[4]=90 ,*(pArray+4)=90

---iArray1 と iArray2 を等値比較(==演算子)---
 iArray1:iArray2 = 等しくない

---iArray1 と pArray を等値比較(==演算子)---
 iArray1:pArray = 等しい

---iArray2 と pArray を等値比較(==演算子)---
 iArray2:pArray = 等しくない

---iArray1 と iArray2 を等価比較(memcmp)---
 iArray1:iArray2 = 等しい

---iArray1 と pArray を等価比較(memcmp)---
 iArray1:pArray = 等しい

---iArray2 と pArray を等価比較(memcmp)---
 iArray2:pArray = 等しい
```

- 実行結果から、以下の点を確認すること
  - iArray1,iArray2,pArray の値が同じであること
  - 値が同じであってもアドレス番地が異なっていると等値比較では等しくないと判断されること
  - 等値比較と等価比較の違い

<参考>

- テキスト p.382 等値判定と等価判定

Question10_07

試してみましょう！
memset 関数を配列の初期化をしてみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_07
  ファイル名:Question10_07.c
- 要素数 10 の char 型配列を宣言すること
- 下記の処理を行うこと
  1. char 型配列を宣言
  2. memset を使って、配列要素を'0'で初期化
  3. 配列要素を表示
  4. memset を使って、配列要素 4 番目~6 番目を'1'で初期化
  5. 配列要素を表示
  6. memset を使って、配列要素 8 番目~9 番目を'A'で初期化
  7. 配列要素を表示
- memset で指定するバイト数は、sizeof 演算子を使って算出すること
- 下記の実行結果を表示するコードを記述すること
  実行結果

```
---buf を 0 で初期化---
buf[0]=0 buf[1]=0 buf[2]=0 buf[3]=0 buf[4]=0
buf[5]=0 buf[6]=0 buf[7]=0 buf[8]=0 buf[9]=0

---buf+3 から 3 文字を 1 で初期化---
buf[0]=0 buf[1]=0 buf[2]=0 buf[3]=1 buf[4]=1
buf[5]=1 buf[6]=0 buf[7]=0 buf[8]=0 buf[9]=0

---buf+7 から 2 文字を A で初期化---
buf[0]=0 buf[1]=0 buf[2]=0 buf[3]=1 buf[4]=1
buf[5]=1 buf[6]=0 buf[7]=A buf[8]=A buf[9]=0
```

Question10_08

試してみましょう！
malloc 関数でメモリを確保し、free 関数で解放してみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_08
  ファイル名:Question10_08.c
- ヒープ領域上に int 型の領域を 500 個分確保すること
- メモリの確保サイズは、sizeof 演算子を使って算出すること
- 下記の処理を行うこと
  1. malloc 関数を使って、メモリを確保する
  2. 確保したメモリに 0～500 の値を設定する
  3. メモリの内容の一部を表示する
  4. メモリを解放する
- 下記の実行結果を表示するコードを記述すること
  実行結果

```
pArray[490] = 490
pArray[491] = 491
pArray[492] = 492
pArray[493] = 493
pArray[494] = 494
pArray[495] = 495
pArray[496] = 496
pArray[497] = 497
pArray[498] = 498
pArray[499] = 499
```

- 実行結果から、値が正常に格納されていることを確認する

Question10_09

九九を格納する領域を malloc 関数でメモリ上に確保し、2 つの数字をかけた値をメモリから取り出すプログラムを作ってみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_09
  ファイル名:Question10_09.c
- ヒープ領域上に int 型の領域を 81 個分確保すること
- メモリの確保サイズは、sizeof 演算子を使って算出すること
- 下記の処理を行うこと
  1. malloc 関数を使って、メモリ領域を確保する
  2. 確保したメモリ領域に九九の値を格納する
  3. 数字の入力を促し、2 つの入力値を受け取る
  4. 2 つの入力値から、メモリの場所を算出する
  5. メモリの場所と九九の値を表示する
- 九九の値を表示するときは、for 文を利用せず、該当するアドレス番地から値を取り出して表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値、実行時により変化)

```
2 つの数字をかけた値をメモリから取り出して表示します。
1 つ目の数字 1～9 の数字を入力してください: 1
2 つ目の数字 1～9 の数字を入力してください: 1
アドレス番地: 01705E68+0
1×1=1
```

```
2 つの数字をかけた値をメモリから取り出して表示します。
1 つ目の数字 1～9 の数字を入力してください: 5
2 つ目の数字 1～9 の数字を入力してください: 5
アドレス番地: 01165E68+40
5×5=25
```

```
2 つの数字をかけた値をメモリから取り出して表示します。
1 つ目の数字 1～9 の数字を入力してください: 9
2 つ目の数字 1～9 の数字を入力してください: 9
アドレス番地: 00818260+80
9×9=81
```

Question10_10

入力値に応じてメモリを動的に確保するプログラムを作成しましょう。
ポインタのポインタを利用して、入力された文字列へのポインタを格納する領域、入力された文字列を格納する領域を扱えるようにしましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q10
  プロジェクト名:Question10_10
  ファイル名:Question10_10.c
- main 関数で下記の処理を行うこと
  1. char 型の配列と char 型のポインタのポインタを宣言
  2. 入力する行数の入力を促す
  3. 入力する行数×char 型へのポインタ変数サイズのメモリ領域を確保
  4. 確保したメモリを初期化
  5. 以下の処理を入力する行数回繰り返す
     - 文字列の入力を促し、文字列を受け取る
     - 受け取った文字列の長さ×char 型変数のサイズのメモリを確保
     - 確保したメモリに受け取った文字列を格納
  6. 文字列を入力された順に表示
  7. 文字列を入力された逆順に表示
  8. メモリを解放
- メモリの確保は malloc 関数を使い、解放は free 関数を使うこと
- メモリの確保サイズは、sizeof 演算子を使って算出すること
- 変数名、変数の型は必要に応じて決定し宣言すること
- メモリ確保の状態を確認するために、アドレス番地を表示すること
- 次ページの実行結果を表示するコードを記述すること

実行結果(赤字:入力値、太字:入力値、実行時により変化)

```
入力する行数を入れてください: 7

------------------------------
アドレス番地 ppmem:00B48EC8
アドレス番地*ppmem:00000000
確保サイズ:28
------------------------------
1 行目 #: Monday
2 行目 #: Tuesday
3 行目 #: Wednesday
4 行目 #: Thursday
5 行目 #: Friday
6 行目 #: Saturday
7 行目 #: Sunday

------------------------------
入力された値の表示 入力順
1 行目 Monday     3 番目の文字:n  : メモリのサイズ:7      :アドレス番地:00B38D78
2 行目 Tuesday    3 番目の文字:e  : メモリのサイズ:8      :アドレス番地:00B38D18
3 行目 Wednesday  3 番目の文字:d  : メモリのサイズ:10     :アドレス番地:00B39C88
4 行目 Thursday   3 番目の文字:u  : メモリのサイズ:9      :アドレス番地:00B39DB8
5 行目 Friday     3 番目の文字:i  : メモリのサイズ:7      :アドレス番地:00B39EA8
6 行目 Saturday   3 番目の文字:t  : メモリのサイズ:9      :アドレス番地:00B39CC8
7 行目 Sunday     3 番目の文字:n  : メモリのサイズ:7      :アドレス番地:00B39EB8
------------------------------
入力された値の表示 逆順
7 行目 Sunday     4 番目の文字:d  : メモリのサイズ:7      :アドレス番地:00B39EB8
6 行目 Saturday   4 番目の文字:u  : メモリのサイズ:9      :アドレス番地:00B39CC8
5 行目 Friday     4 番目の文字:d  : メモリのサイズ:7      :アドレス番地:00B39EA8
4 行目 Thursday   4 番目の文字:r  : メモリのサイズ:9      :アドレス番地:00B39DB8
3 行目 Wednesday  4 番目の文字:n  : メモリのサイズ:10     :アドレス番地:00B39C88
2 行目 Tuesday    4 番目の文字:s  : メモリのサイズ:8      :アドレス番地:00B38D18
1 行目 Monday     4 番目の文字:d  : メモリのサイズ:7      :アドレス番地:00B38D78
------------------------------
```

※○番目の文字は、動作を確認するために表示しています。どうすれば表示できるかを考えましょう。

11. 文字列操作

Question11_01

試してみましょう！
malloc 関数で確保したメモリ領域に文字列を格納し、strlen 関数で長さ文字数を取得するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_01
  ファイル名:Question11_01.c

- 以下の処理の流れで作成すること
  1. メモリ領域を確保する

     strlen 関数で文字数を取得する

     文字列と文字数を表示する

     取得した文字数を使って、文字列に文字'!'と'¥0'を追加する

     文字を追加した後の文字列と文字数を表示する

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
文字列 HelloWorld の長さ 10
----文字を追加----
文字列 HelloWorld!の長さ 11
```

Question11_02

入力値を文字列型で受け取り、strlen 関数で長さを取得するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_02
  ファイル名:Question11_02.c

- 以下の処理の流れで作成すること
  1. 文字列の入力を促す

     入力値を受け取る

     strlen 関数を呼び出し文字列の長さを取得する

     入力値と文字数を表示する

- 下記の実行結果を表示するコードを記述すること
  実行結果(赤字:入力値、太字:入力値により変化)

```
文字列を入力してください:abcdefg

入力された文字列: abcdefg
文字列の長さ: 7
```

Question11_03

試してみましょう！
strcmp 関数を利用して、構造体メンバと文字列を比較してみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_03
  ファイル名:Question11_03.c

- 下記のように構造体と文字列を宣言、初期化して利用すること

```
typedef struct {
      char str[256];
}Frog;

Frog frog = { "井の中の蛙、大海を知らず" };

char check[256] = "井の中の蛙、大海を知らず";
```

- strcmp 関数を利用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
文字列は同じです
"井の中の蛙、大海を知らず"はことわざです
```

Question11_04

試してみましょう！
strcpy 関数を利用して、文字列をコピーしてみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_04
  ファイル名:Question11_04.c

- 下記のように文字列を宣言、初期化して利用すること

```
char str1[256] = "コピー未完了";
char str2[256] = "コピー完了";
```

- strcpy 関数を利用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
----コピー前----
コピー元 : コピー完了
コピー先 : コピー未完了

内容をコピーします

----コピー後----
コピー元 : コピー完了
コピー先 : コピー完了
```

- 実行結果より、コピー先の文字列がコピー元の文字列と同じになっていることを確認する

Question11_05

試してみましょう！
strcat 関数を利用して、文字列を連結してみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_05
  ファイル名:Question11_05.c

- 下記のように文字列を宣言、初期化して利用すること

```
char str1[256] = "ありがとう";
char str2[256] = "ございました";
```

- strcat 関数を利用すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
----連結前----
連結先の文字列(str1)： ありがとう
連結する文字列(str2)： ございました

文字列の連結を行います

----連結後----
連結後の文字列(str1)： ありがとうございました
連結する文字列(str2)： ございました
```

- 以下の点を確認すること
  - 変数 str1 に変数 str2 の文字列が連結し、str1 が変化していること
  - 変数 str2 は変化していないこと

Question11_06

試してみましょう！
sprintf 関数で数字や記号と組み合せた文字列を取得するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_06
  ファイル名:Question11_06.c

- char 型の配列を下のように 4 つ宣言し、1 つを”HelloWorld”で初期化すること

```
//char 型の配列を宣言
char str[256] = "HelloWorld";
char resultstr1[256];
char resultstr2[256];
char resultstr3[256];
```

- sprintf 関数、strlen 関数を組み合せて表示する文字列を作成すること

- printf 関数を使って、下記の実行結果を表示するコードを記述すること
  実行結果

```
文字列 HelloWorld の長さ 10
----文字を追加----
文字列 HelloWorld!の長さ 11
```

Question11_07

試してみましょう！
scanf 関数を使い、入力値が偶数か、奇数かを表示するプログラムを作りましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_07
  ファイル名:Question11_07.c

- 入力値は、整数型で受け取ること

- 下記の実行結果になること(赤字:入力値、太字:入力値により変化)
  実行結果 1: 奇数の場合

```
整数を入力してください。
123
123 は奇数です。
```

実行結果 2: 偶数の場合

```
整数を入力してください。
456
456 は偶数です。
```

Question11_08

試してみましょう！
Question05_06 で作成したプログラムは、入力値を文字列で受け取り、整数に変換していました。scanf で入力値を整数型で受け取るようにプログラムを変更しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_08
  ファイル名:Question11_08.c

- Question05_06.c をコピーして、上記のファイル名で保存すること

- 入力値は、整数型で受け取ること

- for 文を 1 つ利用すること

- 下記の実行結果になること
  実行結果(赤字:入力値)

```
入力された段の九九を表示します
1～9 の数値を入力してください
5
5 の段   5   10   15   20   25   30   35   40   45
```

Question11_09

試してみましょう！
Question05_09 は入力値を文字列で受け取り、整数に変換していました。scanf で入力値を整数型で受け取るようにプログラムを変更しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_09
  ファイル名:Question11_09.c

- Question05_09.c をコピーして、上記のファイル名で保存すること

- for 文を 2 つ利用し、for 文をネストして利用すること

- 入力値は scanf を 1 度だけ使い、整数で受け取ること

- printf 文は、プログラム全体で 5 つ利用すること

- 下記の実行結果になること
  実行結果(赤字:入力値、太字:入力値により変化)

```
縦と横の数を 1～9 の値を入力してください
カンマ区切りで 2 つの値を入力してください:4,9
縦:4　横:9 の四角形を表示します。
*********
*********
*********
*********
```

Question11_10

Question11_09 のプログラムを改良して、2 つの文字を交互に表示し四角形を描くプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_10
  ファイル名:Question11_10.c

- Question11_09.c をコピーして、上記のファイル名で保存すること

- for 文を 2 つ利用し、for 文をネストして利用すること
-
- for 文内に if 文を追加し、記号が交互に表示すること

- 入力値は scanf を 1 度だけ使い、整数値 2 つと文字型 2 つで受け取ること

- 下記の実行結果になること
  実行結果(赤字:入力値、太字:入力値により変化)

```
縦と横の数、表示する記号を 2 種類入力すると、四角形を表示します。
縦の数、横の数、記号を次の形式で入力してください
入力形式: [1～9],[1～9],[@#$&*-],[@#$&*-]
入力例: 4,9,@,-

5,8,$,-

縦: 5  横: 8 の四角形を表示します。
$-$-$-$-
-$-$-$-$
$-$-$-$-
-$-$-$-$
$-$-$-$-
```

Question11_11

英字文字列の大文字、小文字を逆転させるプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_11
  ファイル名:Question11_11.c

- 下記の関数を作成すること
  - 関数名: changeReverse
  - 引数: 文字型のポインタ
  - 戻り値: なし
  - 処理内容: 文字が英小文字の場合英大文字、英大文字の場合英小文字に変換する

- 入力値は scanf を 1 度だけ使い、文字型配列で受け取ること

- 下記の実行結果になること
  実行結果(赤字:入力値、太字:入力値により変化)

```
入力文字列のアルファベットを大文字→小文字、小文字→大文字に変換します
英字文字列を入力してください
abc123ABC456
ABC123abc456
```

Question11_12

Question03_16 で作成した身長と体重を入力して、BMI と適正体重を算出するプログラムは、整数値で身長と体重を入力していました。
小数点値での入力を受け付けるように変更しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_12
  ファイル名:Question11_12.c

- BMI と適正体重は、下記の算出式で求めること
  - BMI の算出式:体重(kg)÷身長(m)^2
  - 適正体重の算出式:身長(m)^2×22
- 身長は、cm 単位の数での入力を促し、浮動小数点型で受け取ること
- 体重は、kg 単位の数での入力を促し、浮動小数点型で受け取ること
- BMI 値、適正体重は、小数点第 1 位まで表示すること
  実行結果 (赤字:入力値 太字:入力値により変化)

```
身長(cm)を入力してください。:167.5
体重(kg)を入力してください。:51.9
身長: 167.5cm 体重: 51.9kg
BMI:18.5
適正体重:61.7kg
```

<ヒント>

- scanf で double 型を扱う場合は、%lf、float 型を扱う場合は、%f を使う

Question11_13

Question09_07 で作成した円周と円の面積を計算するプログラムを変更して、半径を入力し、メニューを表示するプログラムに変更しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_13
  ファイル名:Question11_13.c
- 以下の関数を追加すること
  - 関数名: changeLowertoUpper
  - 引数:文字型ポインタ
  - 戻り値: 戻り値なし
  - 処理内容: 英小文字を英大文字に変換する

  - 関数名: makeStringLength
  - 引数:文字型ポインタ,double 型のポインタ
  - 戻り値: 戻り値なし
  - 処理内容: 円周表示用の文字列を作る

  - 関数名: makeStringArea
  - 引数:文字型ポインタ,double 型のポインタ
  - 戻り値: 戻り値なし
  - 処理内容: 円の面積表示用の文字列を作る

  - 関数名: makeStringLengthArea
  - 引数:文字型ポインタ,double 型のポインタ, double 型のポインタ
  - 戻り値: 戻り値なし
  - 処理内容: 円周表示用処理、円の面積表示用処理を呼び文字列を作る

次ページに続く

- main 関数は、以下の処理を行うこと
  1. 円周と円の面積を格納する変数を準備

     半径の入力を受け付ける

     メニューを表示

     メニューの入力値を英大文字に変換する関数に渡す

     メニューに合わせた文字が入力されるまで下記の処理を繰り返す

     ・入力値が'A':円周算出関数、円の面積関数を呼び、表示文字列を生成

     ・入力値が'B':円周算出関数を呼び、表示文字列を生成

     ・入力値が'C':円の面積関数を呼び、表示文字列を生成

     ・入力値が A、B、C 以外の文字:再度入力を促すメッセージを表示し 3 に戻る

     表示文字列を表示
- 表示文字列は、sprintf 関数を使い生成すること
- 値の入力は、scanf 関数を 2 箇所で使い値を受け取ること
- ※ 連続で scanf を使うと改行文字が入ってしまうため、代入抑止処理が必要
  代入抑止処理の書き方例　scanf("%*c");
- 半径の入力値は、浮動小数点型、メニューの入力値は、文字型で受け取ること
- 以下の実行結果を表示するコードを記述すること
- (赤字:入力値、太字:入力値により変化)

実行結果 1: メニューA もしくは a

```
---円周率、円の面積を計算するプログラム---
半径を小数点第 1 位までの数値で入力してください
半径: 5.6

メニューの文字を入力してください
A or a: 円周と円の面積を表示
B or b: 円周を表示
C or c: 円の面積を表示
メニューの文字:
a

---円周を表示します---
円周: 35.17

---円の面積を表示します---
円の面積: 98.47
```

次ページに続く

実行結果 2: メニューB もしくは b

```
---円周率、円の面積を計算するプログラム---
半径を小数点第１位までの数値で入力してください
半径: 5.6

メニューの文字を入力してください
A or a: 円周と円の面積を表示
B or b: 円周を表示
C or c: 円の面積を表示
メニューの文字:
b

---円周を表示します---
円周: 35.17
```

実行結果 3: メニューC もしくは c

```
---円周率、円の面積を計算するプログラム---
半径を小数点第１位までの数値で入力してください
半径: 5.6

メニューの文字を入力してください
A or a: 円周と円の面積を表示
B or b: 円周を表示
C or c: 円の面積を表示
メニューの文字:
c

---円の面積を表示します---
円の面積: 98.47
```

実行結果 4: A、B、C もしくは、a,b,c 以外

```
---円周率、円の面積を計算するプログラム---
半径を小数点第１位までの数値で入力してください
半径: 5.6

メニューの文字を入力してください
A or a: 円周と円の面積を表示
B or b: 円周を表示
C or c: 円の面積を表示
メニューの文字:
d

もう一度入力してください

メニューの文字を入力してください
A or a: 円周と円の面積を表示
B or b: 円周を表示
C or c: 円の面積を表示
メニューの文字:
```

Question11_14

Question07_03 で作成した、配列を使用して 10 個の数字を並べ替えるプログラムを編集して、20 個以内の数字を並べ変えるプログラムを作成しましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_14
  ファイル名:Question11_14.c

- 20 個の整数値を格納する配列を使用すること
- 下記の関数を作成すること
  - 関数名: originalSort
  - 引数: int 型配列　並べ替え対象、int 型　配列のサイズ
  - 戻り値: なし
  - 処理内容: int 型配列に格納されている値を並べ替える

  - 関数名: showArray
  - 引数: int 型配列　表示対象、int 型　配列のサイズ
  - 戻り値: なし
  - 処理内容: int 型配列に格納されている値を表示する

- main 関数で以下の処理を行うこと
  1: 必要な変数を宣言
  2: 入力を促すメッセージを表示
  3: 20 個もしくは-1 が入力されるまで、整数型で値を受け取り配列に格納
  4: 関数 showArray を呼び、並べ替え前の状態を表示
  5: 関数 originalSort を呼び、数字を並べ替える
  6: 関数 showArray を呼び、並べ替え後の状態を表示
- scanf 関数の第二引数には int 型配列要素へのポインタを指定すること
- 次ページの実行結果を表示するコードを記述すること
- (赤字:入力値、太字:入力値により変化)

実行結果 1: 数字を 20 個入れた場合

```
---正の整数値を並べ替えます---
正の整数値を入力してください
-1 を入力する、もしくは 20 個に到達すると入力受け付けを終了します
1 番目の数:12
2 番目の数:23
3 番目の数:34
4 番目の数:90
5 番目の数:98
6 番目の数:97
7 番目の数:74
8 番目の数:71
9 番目の数:45
10 番目の数:56
11 番目の数:72
12 番目の数:22
13 番目の数:44
14 番目の数:11
15 番目の数:0
16 番目の数:26
17 番目の数:29
18 番目の数:83
19 番目の数:82
20 番目の数:91
並べ替え前の状態：
 12  23  34  90  98  97  74  71  45  56  72  22  44  11  0  26  29  83  82  91
並べ替え後の状態：
 0  11  12  22  23  26  29  34  44  45  56  71  72  74  82  83  90  91  97  98
```

実行結果 2: -1 を入れた場合

```
---正の整数値を並べ替えます---
正の整数値を入力してください
-1 を入力する、もしくは 20 個に到達すると入力受け付けを終了します
1 番目の数:34
2 番目の数:45
3 番目の数:76
4 番目の数:12
5 番目の数:23
6 番目の数:48
7 番目の数:90
8 番目の数:1
9 番目の数:-1
並べ替え前の状態：
 34  45  76  12  23  48  90  1
並べ替え後の状態：
 1  12  23  34  45  48  76  90
```

Question11_15

試してみましょう！
コマンドライン引数の扱われ方をプログラムでメモリの値などの情報を表示して確認してみましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_15
  ファイル名:Question11_15.c

- main 関数内で、コマンドライン引数を受け取るように宣言すること

- main 関数で受け取った引数を下記のようにプレースホルダを使い表示すること
  - “第 1 引数:args=%d”,args
  - “第 2 引数:argv の値 アドレス番地(argv)=%p”
  - “第 2 引数:*argv の値 アドレス番地(*argv)=%p”

- for 文を使い、コマンドライン引数に関する値を下記のようにプレースホルダを使い表示すること
  - “アドレス番地*(argv+%d)=%p, argv[%d]=%p”
  - "値: argv[%d]=%s“
  - “長さ: strlen(argv[%d])=%d”

- コマンドラインで以下のように実行すること(赤字:実行時に入力する値)

```
Question11_15 A BC DEF GHIL MNOPQ
```

- 次ページの実行結果を表示するコードを記述すること

実行結果(太字:環境、実行時により変化)

```
<<< main 関数の引数 >>>
第 1 引数:args=6
第 2 引数:argv の値 アドレス番地(argv)=00D19AD0
第 2 引数:*argv の値 アドレス番地(*argv)=00D19AEC

<<< コマンドライン引数の表示 >>>
---0 番目---
アドレス番地: *(argv+0)=00D19AEC,  argv[0]=00D19AEC
値: argv[0]=Question11_15
長さ: strlen(argv[0])=13
---1 番目---
アドレス番地: *(argv+1)=00D19AFA,  argv[1]=00D19AFA
値: argv[1]=A
長さ: strlen(argv[1])=1
---2 番目---
アドレス番地: *(argv+2)=00D19AFC,  argv[2]=00D19AFC
値: argv[2]=BC
長さ: strlen(argv[2])=2
---3 番目---
アドレス番地: *(argv+3)=00D19AFF,  argv[3]=00D19AFF
値: argv[3]=DEF
長さ: strlen(argv[3])=3
---4 番目---
アドレス番地: *(argv+4)=00D19B03,  argv[4]=00D19B03
値: argv[4]=GHIL
長さ: strlen(argv[4])=4
---5 番目---
アドレス番地: *(argv+5)=00D19B08,  argv[5]=00D19B08
値: argv[5]=MNOPQ
長さ: strlen(argv[5])=5
```

14 バイト
2 バイト
3 バイト
4 バイト
5 バイト

- 実行結果から、以下の点を確認すること
  - main 関数の第 2 引数の値が、コマンドライン引数 0 番目のアドレス番地と同じであること
  - コマンドライン引数 0 番目と 1 番目の差、1 番目と 2 番目のアドレス番地の差が、それぞれコマンドライン引数に入力された文字列の長さ+1(終端文字分)になっていること
  - コマンドライン引数の以下 2 つの方法で、同じアドレス番地を指していること
    - *(argv+添え字番号)と指定する場合と、
    - argv[添え字番号]と指定する場合
  - argv [添え字番号]と指定すると、コマンドライン引数に入力した値が表示できること

Question11_16

Question11_11 で作成した英字文字列の大文字、小文字を逆転させるプログラムを変更して、コマンドライン引数に入力された文字列を変換しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_16
  ファイル名:Question11_16.c

- Question11_11.c をコピーして上記のファイル名に変更すること

- 1 番目のコマンドライン引数を char 型のポインタに格納すること
-
- changeReverse 関数にコマンドライン引数の情報を渡すこと

- コマンドラインで以下のように実行すること(赤字: 実行時に入力する値)

```
Question11_16 abc123ABC456
```

- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:コマンドライン引数により変化)

```
コマンドライン引数の値を扱います
英大文字→英小文字、英小文字→英大文字に変換します

変換前:abc123ABC456
変換後:ABC123abc456
```

Question11_17

Question10_03 で、配列要素の値を前から順、後から順に表示する処理を作成しました。これを応用して、コマンドライン引数の値を前から順と後ろから順に表示するプログラムを作りましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_17
  ファイル名:Question11_17.c

- コマンドラインで以下のように 1 文字ずつ空白を開けて、値を入力し実行すること(赤字: 実行時に入力する値)

```
Question11_17 A B C D E F G
```

- 下記の実行結果を表示するコードを記述すること
  実行結果(太字:環境、実行タイミングにより変化)

```
前から順:A B C D E F G
後から順:G F E D C B A
```

Question11_18

コマンドライン引数に、数字、英大文字、英小文字、空白文字、その他がそれぞれ何個ずつあるかを数えるプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_18
  ファイル名:Question11_18.c

- 複数のコマンドライン引数を連結して、char 型のポインタに格納すること

- 数字、英大文字、英小文字、空白文字、その他の数を格納する int 型配列を用意し文字を数え、全体の文字列に対して、何パーセントあるかを表示すること

- コマンドラインで以下のように実行すること(赤字: 実行時に入力する値)

```
Question11_18 Hello ! I'm grad to meet you. 12/03/20
```

- コマンドライン引数に何も指定されなかった場合は、引数がなかったことを示すメッセージを表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果 コマンドライン引数あり(太字:コマンドライン引数により変化)

```
コマンドライン引数に数字、英大文字、英小文字、空白文字、その他がそれぞれ何個ずつあるかを表示します

コマンドライン文字列:Hello ! I'm grad to meet you. 12/03/20
全文字数:39
数字　　:　 6　　15.4%
英大文字:　 2　　 5.1%
英小文字:　18　　46.2%
空白文字:　 8　　20.5%
その他　:　 5　　12.8%
```

  実行結果 コマンドライン引数なし

```
コマンドライン引数に文字列がありません
```

Question11_19

Question11_12 で作成した身長と体重を入力して、BMI と適正体重を算出するプログラムを変更して、コマンドライン引数で身長と体重を受け取れるようにしましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_19
  ファイル名:Question11_19.c
- BMI と適正体重は、下記の算出式で求めること
  - BMI の算出式:体重(kg)÷身長(m)^2
  - 適正体重の算出式:身長(m)^2×22

- コマンドライン引数で以下の値を受け取ること
  - 値 1: 身長(cm)
  - 値 2: 体重(kg)
- コマンドラインで以下のように 3 つのパターンで実行できること
  (赤字: 実行時に入力する値)

  実行方法 1:
  ```
  Question11_18 167.5 51.8
  ```

  実行方法 2:
  ```
  Question11_18 167.5
  ```

  実行方法 3:
  ```
  Question11_18
  ```
- main 関数は、以下の処理を行うこと
  1. コマンドライン引数の数をチェックする
     引数が 3 つの場合:
     身長と体重をコマンドライン引数から受け取る
     引数が 2 つの場合:
     身長をコマンドライン引数から受け取り、体重の入力を促す
     引数が 2 つもしくは 3 つ以外の場合:
     身長と体重の入力を促す
  2. 取得し値を使い BMI と適正体重を算出する
  3. 計算結果を表示する

次ページへ続く

実行結果 (赤字:入力値 太字:入力値により変化)

コマンドライン引数が 3 つの場合(Question11_19 167.5 51.8)

```
身長: 167.5cm 体重: 51.9kg
BMI:18.5
適正体重:61.7kg
```

コマンドライン引数が 2 つの場合(Question11_19 167.5)

```
体重(kg)を入力してください。: 51.8
身長 : 167.5cm 体重: 51.8kg
BMI:18.5
適正体重:61.7kg
```

コマンドライン引数が 2 つもしくは 3 つ以外の場合(Question11_19)

```
身長(cm)を入力してください。:167.5
体重(kg)を入力してください。:51.9
身長: 167.5cm 体重: 51.9kg
BMI:18.5
適正体重:61.7kg
```

<ヒント>

- scanf で double 型を扱う場合は、%lf、float 型を扱う場合は、%f を使う
- 文字列を浮動小数点型に変える場合は、atof 関数を使う

Question11_20

Question09_08 で作成したプログラムを変更して、コマンドライン引数から、お弁当名、本体価格、持ち帰りかイートインかを読み取ることができるようにしましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q11
  プロジェクト名:Question11_20
  ファイル名:Question11_20.c
- Question09_08.c をコピーして上記の名前に変更すること。
- 構造体にお弁当名を追加した以下の構造体を定義すること
  - char 型配列 お弁当名
  - int 型　本体価格
  - int 型　8%の消費税額
  - int 型　10%の消費税額
  - int 型　8%消費税額込みの価格
  - int 型　10%消費税額込みの価格

```
typedef struct {
        char name[256];
        int price;
        int tax8Price;
        int tax10Price;
        int price8in;
        int price10in;
}tagLunchBox;
```

- お弁当情報を初期化する initLunchBoxData 関数を以下のように変更すること
  - 関数名: initLunchBoxData
  - 引数 1:tagLunchBox 型　お弁当価格用構造体
  - 戻り値:なし
  - 処理内容:お弁当の本体価格から、消費税額、消費税込み価格を算出し、構造体メンバを初期化する。消費税額の算出は、tacCalc 関数を呼び出す

- お弁当価格を表示する showPrice 関数を以下のように変更すること
  - 関数名: showPrice
  - 引数 1: tagLunchBox 型　お弁当価格用構造体
  - 引数 2:int 型 持ち帰りの場合 0、イートインの場合 1
  - 戻り値:なし
  - 処理内容:お弁当名を表示後、持ち帰りかイートインかを判断し、条件に合った値を構造体メンバから取り出し表示する

次ページに続く

- コマンドライン引数で以下の値を受け取ること
  - 値 1: お弁当名を入力
  - 値 2: 価格を半角数字で入力
  - 値 3: out もしくは in と入力 :out-持ち帰り、in-イートイン
- コマンドラインで以下のように実行すること(赤字: 実行時に入力する値)

```
Question11_19 のり弁当 350 in
```

- main 関数は、以下の処理を行うこと
  1. コマンドライン引数の数が 4(プログラム名+引数 3 つ)かをチェックする
     引数が 4 つではない場合: 使い方説明を表示しプログラム終了
     引数が 4 つの場合: 2 以降の処理を行う
     構造体 tagLunchBox の変数を宣言する
     コマンドライン引数から、strcpy 関数を使用し構造体メンバにお弁当名を格納する
     コマンドライン引数から、atoi 関数を使用し構造体メンバに価格を格納する
     構造体のポインタを initLunchBoxPrice 関数に渡す
     イートインか持ち帰りかを判断する int 型変数を宣言する
     strcmp 関数を使用し、コマンドライン引数から持ち帰りかイートインかを判断する
     持ち帰りの場合: 0、イートインの場合: 1
     どちらにも判別できなかった場合:入力を促す
     構造体のポインタと持ち帰りかイートインかの入力値を showPrice 関数に渡す
- 消費税率は、グローバルで定数として宣言すること
- taxCalc 関数は、main 関数からではなく、initLunchBoxPrice 関数からのみ呼び出すこと
- 次ページの実行結果を表示するコードを記述すること

実行結果(赤字:入力値、太字:入力値により変化)

コマンドライン引数が 3 つなかった場合 (Question11_20)

```
以下の形式でプログラムを動かしてください
Usage : Question11_19 値 1 値 2 値 3
値 1: お弁当名を入力
値 2: 価格を半角数字で入力
値 3: out もしくは in と入力 :out-持ち帰り、in-イートイン
```

持ち帰り指定の場合 (Question11_20 のり弁当 350 out)

```
---- のり弁当 の価格 ----
持ち帰りの税率: 8%
税込み(8%)価格 : 378 円
消費税額(8%)   : 28 円
```

イートイン指定の場合 (Question11_20 のり弁当 350 in)

```
---- のり弁当 の価格 ----
イートインの税率: 10%
税込み(10%)価格 : 385 円
消費税額(10%)   : 35 円
```

持ち帰りかイートインが読み取れなかった場合で、持ち帰り選択
(Question11_20 のり弁当 350 a)

```
持ち帰りかイートインかが読み取れませんでした。
持ち帰りの場合:0、イートインの場合:1 を入力してください。
0
---- のり弁当 の価格 ----
持ち帰りの税率: 8%
税込み(8%)価格 : 378 円
消費税額(8%)   : 28 円
```

持ち帰りかイートインが読み取れなかった場合で、イートイン選択
(Question11_20 のり弁当 350 a)

```
持ち帰りかイートインかが読み取れませんでした。
持ち帰りの場合:0、イートインの場合:1 を入力してください。
1
---- のり弁当 の価格 ----
イートインの税率: 10%
税込み(10%)価格 : 385 円
消費税額(10%)   : 35 円
```

12. 複数のファイルによる開発

Question12_01

試してみましょう！
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>
- ソースファイルの文字コードを「UTF-8」にし、開発者用コマンドプロンプトでコンパイルを実行すること
- ソースファイルと同階層に「bin」フォルダを作成し、コンパイル先に「bin」フォルダを指定すること
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q12
  プロジェクト名:Question12_01
  ファイル名:Question12_01_main.c、Question12_01_sub.c
- main 関数を持つファイルと、sub 関数を持つファイルと 2 つのソースファイルを用意すること
- Question12_01_main.c で Question12_01_sub.c をインクルードすること
- Question12_01_sub.c に下記の関数を作成すること
  - 関数名: sub
  - 引数: int 型のポインタ
  - 戻り値: なし
  - 処理内容: 引数で受け取った値に 5 を加える
- Question12_01_main.c に main 関数を作成し、以下の処理を行うこと
  1. int 型の変数を宣言し、10 で初期化
  2. 値の表示
  3. sub 関数に int 型変数のポインタを渡す
  4. sub 関数呼び出し後の値を表示する
- main 関数、sub 関数で処理を行う前後の時点で値を表示すること

- 下記の実行結果を表示するコードを記述すること
  実行結果

```
main: sub 関数呼び出し前 : 10
sub: sub 関数処理前 : 10
sub: sub 関数処理後 : 15
main: sub 関数呼び出し後 : 15
```

Question12_02

試してみましょう！
Question12_01 にヘッダファイルを追加してみましょう。
下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソースファイルの文字コードを「UTF-8」にし、開発者用コマンドプロンプトでコンパイルを実行すること

- ソースファイルと同階層に「bin」フォルダを作成し、コンパイル先に「bin」フォルダを指定すること

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q12
  プロジェクト名:Question12_02
  ファイル名:Question12_02_main.c、Question12_02_sub.c
  ヘッダファイル名: sub.h

- Question12_01_main.c、Question12_01_sub.c をコピーして上記ファイル名に変更すること

- sub.h ヘッダファイルで以下の定義を行うこと

```
#ifndef __SUB_H__
#define __SUB_H__

void sub(int *inum);

#endif
```

- Question12_02_main.c、Question12_02_sub.c で sub.h をインクルードすること

- ヘッダファイル利用前と同じ、下記の実行結果を表示するコードを記述すること
  実行結果

```
main: sub 関数呼び出し前 : 10
sub: sub 関数処理前 : 10
sub: sub 関数処理後 : 15
main: sub 関数呼び出し後 : 15
```

Question12_03

Question11_13 で作成した円周と円の面積を計算する部分と英小文字を英大文字に変換する処理を別のソースファイルとして保存し、ヘッダファイルをインクルードして同様の動作結果となるように編集しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

- ソースファイルの文字コードを「UTF-8」にし、開発者用コマンドプロンプトでコンパイルを実行すること
- ソースファイルと同階層に「bin」フォルダを作成し、コンパイル先に「bin」フォルダを指定すること
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q12
  プロジェクト名:Question12_03
  ファイル名:Question12_03.c、calcircle.c、changechar.c
  ヘッダファイル名: calcircle.h、changechar.c

- Question11_13.c の内容をコピーして上記ファイルを作成すること
- ヘッダファイル calcircle.h で以下の関数定義を行うこと

```
void calcCircleLength(double dRadius,double *circleLength);
void calcCircleArea(double dRadius,double *circleArea);
void makeStringLength(char str[256],double *circleLength);
void makeStringArea(char str[256],double *circleArea);
void makeStringLengthArea(char str[256],double *circleLength,double *circleArea);
```

- ヘッダファイル changechar.h で以下の関数定義を行うこと

```
void changeLowertoUpper(char *cmoji);
```

- Question12_03.c、calcircle.c で calcircle.h をインクルードすること
- Question12_03.c、changechar.c で changechar.h をインクルードすること
- ヘッダファイルに合わせて、calcircle.c と changechar.c を作成すること
- ヘッダファイル利用前と同じ、Question11_13 と同じ実行結果を表示するコードを記述すること

Question12_04

Question12_03 で変更したプログラムを元に、繰り返し算出処理が呼び出せるプログラムを作成しましょう。

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

- ソースファイルの文字コードを「UTF-8」にし、開発者用コマンドプロンプトでコンパイルを実行すること
- ソースファイルと同階層に「bin」フォルダを作成し、コンパイル先に「bin」フォルダを指定すること
- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q12
  プロジェクト名:Question12_04
  ファイル名:Question12_04.c、circleInfo.c、changechar.c
  ヘッダファイル名: circleInfo.h、changechar.h
- 各ファイルで必要に応じて、ヘッダファイルをインクルードすること
- circleinfo.h ヘッダファイルで以下の定義を行うこと

```
typedef struct {
	char lengthTitle[256];
	char areaTitle[256];
	double radius;
	double length;
	double area;
}tagCircleInfo;
void initCircleInfo(tagCircleInfo *circleInfo);
void inputCircleRadius(tagCircleInfo *circleInfo);
void calcCircle(tagCircleInfo *circleInfo);
void printLength(tagCircleInfo *circleInfo);
void printArea(tagCircleInfo *circleInfo);
void printLengthArea(tagCircleInfo *circleInfo);
```

- circleinfo.c ファイルで、以下の関数を作成すること
  - 構造体メンバの初期化処理 initCircleInfo(tagCircleInfo *circleInfo);
  - 半径の入力処理 inputCircleRadius(tagCircleInfo *circleInfo);
  - 円周、円の面積算出処理 calcCircle(tagCircleInfo *circleInfo);
  - 円周表示処理 printLength(tagCircleInfo *circleInfo);
  - 円の面積表示処理 printArea(tagCircleInfo *circleInfo);
  - 円周、円の面積表示処理 printLengthArea(tagCircleInfo *circleInfo);

- コマンドラインで以下のように実行すること(赤字: 実行時に入力する値)
  - 引数あり

```
Question12_04 3.4
```

  - 引数なし

```
Question12_04
```

次ページに続く

- main 関数は、以下の処理を行うこと
  1. コマンドライン引数の数をチェックする

     引数が 2 つ以上場合: コマンドライン引数の値を半径として受け取る

     引数が 2 未満の場合: 半径の入力を促す

     メニューを表示し、入力を受け付ける

     入力文字に合わせて処理を行う
- メニューは下記の内容を表示する

```
メニューの文字を英字で入力してください(大文字小文字区別なし)
A: 円周と円の面積を表示
B: 円周を表示
C: 円の面積を表示
D: 半径の再設定
Q: 終了
```

- メニュー表示後の入力文字は、changechar.c の changeLowertoUpper 関数により小文字を大文字に変換したのち、判断を行うこと
- 関数を活用し、できるだけ類似コードを記載しないようにすること
- 下記の実行結果を表示するコードを記述すること
  (赤字:入力値、太字:入力値により変化)

  実行結果　前半部分: コマンドライン引数あり

```
---円周率、円の面積を計算するプログラム---

---設定されている半径:3.4---
```

  実行結果　前半部分: コマンドライン引数なし

```
---円周率、円の面積を計算するプログラム---
半径を小数点第 1 位までの数値で入力してください
半径: 3.4

---設定されている半径:3.4---
```

  実行結果　メニュー部分

```
メニューの文字を英字で入力してください(大文字小文字区別なし)
A: 円周と円の面積を表示
B: 円周を表示
C: 円の面積を表示
D: 半径の再設定
Q: 終了
メニューの文字:
```

次ページに続く

実行結果 メニュー入力後：文字がＡもしくはａの場合

```
メニューの文字:
a

---円周を表示します---
円周 :21.35

---円の面積を表示します---
円の面積 :36.30
```

実行結果 メニュー入力後：文字がＢもしくはｂの場合

```
メニューの文字:
b

---円周を表示します---
円周 :21.35
```

実行結果 メニュー入力後：文字がＣもしくはｃの場合

```
メニューの文字:
c

---円の面積を表示します---
円の面積 :36.30
```

実行結果 メニュー入力後：文字がＤもしくはｄの場合

```
メニューの文字:
d
半径を小数点第１位までの数値で入力してください
半径: 4.5
```

実行結果 メニュー入力後：文字がＱもしくはｑの場合

```
メニューの文字:
q

終了します
```

実行結果 メニュー入力後：文字がメニューにない文字の場合

```
メニューの文字:
e

文字が違います。もう一度入力してください
```

13. ファイル操作

Question13_01

試してみましょう！

標準入力から 1 文字読み取る機能を持つ getchar 関数を使って、入力した文字をファイルに書き込むプログラムを作成しましょう。

まず、以下の getchar 関数を使ったプログラムを入力し、動作を確認しましょう。

<getchar 関数を使ったプログラム>

```
#include <stdio.h>

int main(void)
{
	int moji;

	while ((moji = getchar()) != EOF)
	{
		printf("取得した文字：%c¥n", moji);
	}

	return 0;
}
```

このプログラムは、画面に文字を入力し「enter」キーを押すと、入力した文字が
画面に表示されます。
プログラムを終了するには、「ctrl」キーを押しながら「Z」キーを押した後、
「enter」キーを押すことで終了します。

getchar 関数を使って、コンソール画面で入力した文字をファイルに書き込むプログラムを作成しましょう。
次ページの<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

・ ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_01
  ファイル名:Question13_01.c

・ main 関数は、以下の処理を行うこと
  1. ファイル名 memo.txt をアクセスモード“w”で開く

     「文字を入力してください」と表示

     getchar 関数を使いコマンドプロンプトで入力した文字を受け取る

     ファイルに書き込む

     Ctrl+Z キーが押されるとファイルを閉じてプログラムを終了する

・ 下記の実行結果を表示するコードを記述すること
  実行結果 コマンドライン引数あり(赤字:入力値)

```
 そ文字を入力してください。
 Ctl+Z を押すと終了します。
 moji input test
 aa
 bb
 cc
 dd
 ee
 ^Z.
```

・ 実行終了後、以下の点を確認すること
  - ファイル名 memo.txt ができていること
  - ファイルの内容が入力した値と一致していること
  - アクセスモードを“ａ”に変えて、コンパイル実行すると、入力内容が memo.txt に追記されること

<ヒント>

・ テキスト p.542~p.543 表 14-2 アクセスモード
・ テキスト p.737 標準入出力関数

Question13_02

試してみましょう！

テキスト p.545~p.546 コード 14-2 を変更して以下 3 つの関数を追加しましょう。
・input 関数 文字の入力、文字数のカウント処理
・write 関数 入力された文字を 1 文字ずつ書き込む処理
・read 関数 文字を読み込み表示する処理

<テキスト p.545~p.546 コード 14-2>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
  FILE* fp;
  char text[] = "sukkriC!";      // 書き込む文字
  int len = strlen(text);        // 文字列の長さを取得
  int ch;

  // 書き込み専用でオープン
  if ((fp = fopen("memo.txt", "w")) == NULL) {
    exit(1);
  }

  for (int i = 0; i < len; i++) {
    fputc(text[i], fp);          /* 配列の要素を 1 文字ずつ書き込む */
  }

  fclose(fp);

  // 読み取り専用でオープン
  if ((fp = fopen("memo.txt", "r")) == NULL) {
    exit(1);
  }
                                 /* ファイルを最後まで読んだらループ終了 */
  while ((ch = fgetc(fp)) != EOF) {
    putchar((char)ch);           // 標準出力(画面)に表示
  }

  fclose(fp);
  return 0;
}
```

次ページに続く

関数作成後は、下記の main 関数になるように機能を分割しましょう。

<関数作成後の main 関数>

```
int main(void) {
  char text[256] = { 0 };
  char filename[256]="memo.txt";
  int len =0;

  printf("main:ファイル名を[英数字 8 文字以内].txt の形式で入力してください¥n");
  scanf("%s", filename);

  input(text, &len);
  write(filename,text, len);
  read(filename);

  return 0;
}
```

下記の<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_02
  ファイル名:Question13_02.c
- main 関数でファイル名の入力を促し、指定されたファイル名に書き込むことができるようにすること
- 指定されたファイル名を読み込み用の関数に渡し、書き込んだ内容を表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果 (赤字:入力値、太字:入力値により変化)

```
main:ファイル名を[英数字 8 文字以内].txt の形式で入力してください
testmemo.txt
----------input----------
input:ファイルに書き込みする文字を入力してください
input:改行を入力すると、ファイルへの書き込みを開始します
testdata-input-2020.03.13
----------write----------
write:ファイルへの書き込みを開始します
write:書き込み文字列:testdata-input-2020.03.13
write:ファイルへの書き込みが完了しました
----------read----------
read:ファイルからの読み込み内容を表示します
testdata-input-2020.03.13
```

Question13_03

試してみましょう！

テキスト p.548~p.549 コード 14-3 を変更して以下 2 つの関数を追加しましょう。
・writeLine 関数 入力された文字列を 1 行ずつ書き込む処理
・readLine 関数 文字列を読み込み表示する処理
決められた文字列をファイルに書き込むのではなく、標準入力で入力された文字列を書き込むように処理を変更しましょう。

<テキスト p.548~p.549 コード 14-3>

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  FILE* fp;
  char wbuf[64];

  // 書き込み専用でオープン
  if ((fp = fopen("memo.txt", "w")) == NULL) {
    exit(1);
  }

  fputs("government of the people, ¥nby the people, ¥nfor the people", fp);

  fclose(fp);

  // 読み取り専用でオープン
  if ((fp = fopen("memo.txt", "r")) == NULL) {
    exit(1);
  }

  while (fgets(wbuf, 64, fp) != NULL) {
    printf("%s", wbuf);    // 標準出力(画面)に表示
  }

  fclose(fp);

  return 0;
}
```

次ページに続く

下記の<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_03
  ファイル名:Question13_03.c
- main 関数でファイル名の入力を促し、指定されたファイル名に書き込むことができるようにすること
- 指定されたファイル名を読み込み用の関数に渡し、書き込んだ内容を表示すること
- 改行コードも含めて、書き込むこと
- 下記の実行結果を表示するコードを記述すること
  実行結果 (赤字:入力値、太字:入力値により変化)

```
ファイル名を[英数字 8 文字以内].txt の形式で入力してください
memomemo.txt
書き込み文字列を 3 行で入力してください。
Hop
Step
Jump
----------ファイルへの書き込みを行います----------
ファイル名:memomemo.txt
----------ファイルへの書き込みが完了しました------
----------ファイルの内容を表示します--------------
ファイル名:memomemo.txt
Hop
Step
Jump
```

<ヒント>

- scanf は、空白文字、改行コードを読み込むことができません
  そのため、標準入力から取得した文字列に改行コードを付加する必要があります。

Question13_04

Question10_10 の複数行入力を受け付けメモリに格納するプログラと,Question13_03 の 1 行ずつファイルに読み書きするプログラムを組み合せて、複数行入力した内容をファイルに書き込むプログラムを作成しましょう。

下記の<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_04
  ファイル名:Question13_04.c
- main 関数でファイル名の入力を促し、指定されたファイル名に書き込むことができるようにすること
- 指定されたファイル名を読み込み用の関数に渡し、書き込んだ内容を表示すること
- 下記の実行結果を表示するコードを記述すること
  実行結果 (赤字:入力値、太字:入力値により変化)

```
ファイル名を[英数字 8 文字以内].txt の形式で入力してください
memomemo.txt
入力する行数を入れてください: 4
1 行目 #: Hop
2 行目 #: Step
3 行目 #: Jump
4 行目 #: Sleep
----------ファイルへの書き込みを行います----------
ファイル名:memomemo.txt
----------ファイルへの書き込みが完了しました------
----------ファイルの内容を表示します--------------
ファイル名:memomemo.txt
Hop
Step
Jump
Sleep
```

Question13_05

下記の<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_05
  ファイル名:Question13_05.c

- main 関数でファイル名の入力を促し、指定されたファイル名に書き込むことができるようにすること
- ファイルに書き込む際、「生年月日」のみ入力チェックを行うこと
  →不正な値がある場合、エラーメッセージを表示し、ファイルへの書き込み、読み込みは実施しない
- 指定されたファイル名を読み込み用の関数に渡し、書き込んだ内容を表示すること
- 入力するデータは、「名前、職業」→文字列、「生年月日」→数値の形式とする

<入力チェック>

・inputCheck という名前の関数を用意すること
・以下の数値を入力した場合、「不正な値が入力されました。システムを終了します。」というエラーメッセージを表示してシステムを終了する
  1. 年→0 以下の数値
  2. 月→0 以下もしくは、13 以上の数値
  3. 日→0 以下もしくは、32 以上の数値

※ bool 型を利用する場合は、stdbool.h をインクルードする必要があります

<ヒント>

- scanf は、空白文字、改行コードを読み込むことができません
  そのため、標準入力から取得した文字列に改行コードを付加する必要があります
- fprintf 関数を使うと、書式付きでファイルへ書き込みをすることができます

・ 下記の実行結果を表示するコードを記述すること
実行結果 (赤字:入力値、太字:入力値により変化)

```
ファイル名を[英数字 8 文字以内].txt の形式で入力してください
employee.txt
名前を入力してください：田中雄二
生年月日はスペース区切りで入力してください: 2022 12 08
職業を入力してください: SE
----------ファイルへの書き込みを行います----------
ファイル名:employee.txt
----------ファイルへの書き込みが完了しました------
----------ファイルの内容を表示します--------------
ファイル名:employee.txt
名前は「田中雄二」
生年月日は「2022 年 12 月 08 日」
職業は「SE」
```

<入力ミスがある場合>
実行結果 (赤字:入力値、太字:入力値により変化)

```
ファイル名を[英数字 8 文字以内].txt の形式で入力してください
employee.txt
名前を入力してください：田中雄二
生年月日はスペース区切りで入力してください: 0 12 08
職業を入力してください: SE
不正な値が入力されました。
システムを終了します。
```

Question13_06

下記の<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q13
  プロジェクト名:Question13_06
  ファイル名:Question13_06.c

- main 関数でファイル名の入力を促し、指定されたファイル名に書き込むことができるようにすること
- 指定されたファイル名を読み込み用の関数に渡し、書き込んだ内容を表示すること
- 改行コードも含めて、書き込むこと
- 下記の実行結果を表示するコードを記述すること
  実行結果 (赤字:入力値、太字:入力値により変化)

```
ファイル名を[英数字 8 文字以内].txt の形式で入力してください
file.txt
書き込み文字列を入力してください。
C 言語
----------ファイルへの書き込みを行います----------
ファイル名:file.txt
----------ファイルへの書き込みが完了しました------
----------ファイルの内容を表示します--------------
ファイル名:file.txt
C 言語
作成したファイルを削除しますか？
yes...1/no...2>1
----------ファイルの削除が完了しました。----------
```

【選択肢を 2 にした場合】

```
<省略>
----------ファイルの内容を表示します--------------
ファイル名:file.txt
C 言語
作成したファイルを削除しますか？
yes...1/no...2>2
システムを終了します。
```

<ヒント>

- scanf は、空白文字、改行コードを読み込むことができません
  そのため、標準入力から取得した文字列に改行コードを付加する必要があります。

14. ビット演算子

Question14_01

試してみましょう！
引数の値を 2 進数表記で表示する以下の関数を使って値を表示してみましょう。

<2 進数で表示する関数:printbi>

```
void printbi(uint8_t n) {
    int i;
    for (i = 8 - 1; i >= 0; i--) {
        printf("%d", (n >> i) & 1);
    }
}
```

※ uint8_t 型を利用するために、stdint.h をインクルードする必要があります

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q14
  プロジェクト名:Question14_01
  ファイル名:Question14_01.c

- 以下のように変数を宣言し、初期化すること

```
uint8_t a =255; // 1111 1111
uint8_t b =197; // 1100 0101
uint8_t c =5;   // 0000 0101
```

- プレースホルダを使って、値を 16 進数、10 進数表記で表示すること

- printbi 関数を使って、値を 2 進数表記で表示すること

- 次ページの実行結果を表示するコードを記述すること

実行結果

```
16 進数、10 進数、2 進数表記で表示します
---a の値---
16 進数:ff
10 進数:255
2 進数:11111111

---b の値---
16 進数:c5
10 進数:197
2 進数:11000101

---c の値---
16 進数:0x05
10 進数:5
2 進数:00000101
```

Question14_02

試してみましょう！
ビット論理演算子を使って算出した結果を表示してみましょう。
結果は、2 進数で表示する関数:printbi を使って値を表示しましょう。

<2 進数で表示する関数:printbi>

```
void printbi(uint8_t n) {
    int i;
    for (i = 8 - 1; i >= 0; i--) {
        printf("%d", (n >> i) & 1);
    }
}
```

※ uint8_t 型を利用するために、stdint.h をインクルードする必要があります

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q14
  プロジェクト名:Question14_02
  ファイル名:Question14_02.c

- 以下のように変数を宣言し、初期化すること

```
uint8_t a =255; // 1111 1111
uint8_t b =197; // 1100 0101
uint8_t c =5;   // 0000 0101
```

- ビット論理演算子を使って計算した結果を表示すること

- printbi 関数を使って、値を 2 進数表記で表示すること

- 次ページの実行結果を表示するコードを記述すること

実行結果

```
---論理積 a & b の結果---
 計算前 a:        11111111
 計算前 b:        11000101
 計算後:         11000101

---論理積 b & c の結果---
 計算前 b:        11000101
 計算前 c:        00000101
 計算後:         00000101

---論理和 a | b の結果---
 計算前 a:        11111111
 計算前 b:        11000101
 計算後:         11111111

---論理和 b | c の結果---
 計算前 b:        11000101
 計算前 c:        00000101
 計算後:         11000101

---排他的論理和 a ^ b の結果---
 計算前 a:        11111111
 計算前 b:        11000101
 計算後:         00111010

---排他的論理和 b ^ c の結果---
 計算前 b:        11000101
 計算前 c:        00000101
 計算後:         11000000

---否定 ~a の結果---
 計算前 a:        11111111
 計算後:         00000000

---否定 ~b の結果---
 計算前 b:        11000101
 計算後:         00111010

---否定 ~c の結果---
 計算前 c:        00000101
 計算後:         11111010
```

- 実行結果表示後、ビット論理演算子の動作を確認しましょう

Question14_03

試してみましょう！
シフト演算子を使って右シフトした結果を表示してみましょう。
結果は、2 進数で表示する関数:printbi を使って値を表示しましょう。

<2 進数で表示する関数:printbi>

```
void printbi(uint8_t n) {
    int i;
    for (i = 8 - 1; i >= 0; i--) {
        printf("%d", (n >> i) & 1);
    }
}
```

※ uint8_t 型を利用するために、stdint.h をインクルードする必要があります

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q14
  プロジェクト名:Question14_03
  ファイル名:Question14_03.c

- 以下のように変数を宣言し、初期化すること

```
uint8_t a =197; // 1100 0101
```

- 右へ 1 ビット、2 ビット、3 ビットシフトした結果を表示すること

- プレースホルダを使って、10 進数表記で表示すること

- printbi 関数を使って、値を 2 進数表記で表示すること

- 次ページの実行結果を表示するコードを記述すること

実行結果

```
---右へ1ビットシフト a>>1 の結果---
 計算前:          10進数: 197   2進数:11000101
 計算後:          10進数:  98   2進数:01100010

---右へ2ビットシフト a>>2 の結果---
 計算前:          10進数: 197   2進数:11000101
 計算後:          10進数:  49   2進数:00110001

---右へ3ビットシフト a>>3 の結果---
 計算前:          10進数: 197   2進数:11000101
 計算後:          10進数:  24   2進数:00011000
```

- 実行結果が以下のようになっていることを確認すること
  - 右へ1ビットシフトした結果
    197÷2の商と結果が同じ
  - 右へ2ビットシフトした結果
    197÷4の商と結果が同じ
  - 右へ3ビットシフトした結果
    197÷8の商と結果が同じ

Question14_04

試してみましょう！
シフト演算子を使って左シフトした結果を表示してみましょう。
結果は、2 進数で表示する関数:printbi を使って値を表示しましょう。

<2 進数で表示する関数:printbi>

```
void printbi(uint8_t n) {
    int i;
    for (i = 8 - 1; i >= 0; i--) {
        printf("%d", (n >> i) & 1);
    }
}
```

※ uint8_t 型を利用するために、stdint.h をインクルードする必要があります

下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q14
  プロジェクト名:Question14_04
  ファイル名:Question14_04.c

- 以下のように変数を宣言し、初期化すること

```
uint8_t a =9; // 0000 1001
```

- 左へ 1 ビット、2 ビット、3 ビットシフトした結果を表示すること

- プレースホルダを使って、10 進数表記で表示すること

- printbi 関数を使って、値を 2 進数表記で表示すること

- 次ページの実行結果を表示するコードを記述すること

実行結果

```
---左へ 1 ビットシフト a<<1 の結果---
 計算前:          10 進数:   9   2 進数:00001001
 計算後:          10 進数:  18   2 進数:00010010

---左へ 2 ビットシフト a<<2 の結果---
 計算前:          10 進数:   9   2 進数:00001001
 計算後:          10 進数:  36   2 進数:00100100

---左へ 3 ビットシフト a<<3 の結果---
 計算前:          10 進数:   9   2 進数:00001001
 計算後:          10 進数:  72   2 進数:01001000
```

- 実行結果が以下のようになっていることを確認すること
  - 左へ 1 ビットシフトした結果
    9×2 と結果が同じ
  - 左へ 2 ビットシフトした結果
    9×4 と結果が同じ
  - 左へ 3 ビットシフトした結果
    9×8 と結果が同じ

15. コーディング規約

Question15_01

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
4 宣言・定義について　(1)禁止事項
- 0 で始まる長さ 2 以上の数字だけの列を定数として使用しない

良い例	悪い例

// 見た目をよくするための桁揃えはしない
a = 0;
b = 10;
c = 100;
	// 見た目をよくするために桁を揃えた例
	a = 000; // 8 進数の 0 と解釈される
	b = 010; // 10 進数の 10 でなく 8 と解釈される
	c = 100; // 10 進数の 100 と解釈される

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{
	const int a = 000;
	const int b = 010;
	const int c = 100;

    int d = 5;

    printf("a*d=%d¥n", a*5);
    printf("b*d=%d¥n", b*5);
    printf("c*d=%d¥n", c*5);

    return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。
<動作結果>

```
a*d=0
b*d=40
c*d=500
```

<期待する結果>

```
a*d=0
b*d=50
c*d=500
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_01
  悪い例：ファイル名:Question15_01_bad.c
  良い例：ファイル名:Question15_01_good.c

- 新たに命令文や変数を追加しないこと

Question15_02

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について (1)型の扱い
・ 演算の型と演算結果の代入先の型が異なる場合は、期待する演算精度の型へキャストしてから演算する

良い例	悪い例

int i1, i2;
long w;
double d;
void func() {
// 浮動小数点型での除算
d = (double)i1 / (double)i2;
// long でのシフト
w = ((long)i1) << i2;
…
}
	int i1, i2;
	long w;
	double d;
	void func() {
	d = i1 / i2; // 整数型での除算
	w = i1 << i2; // int でのシフト
	…
	}

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{
	double d=0;
	int i=22;

	d = 5 / 9*i+32;

	printf("d=%f¥n",d);

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
d=32.000000
```

<期待する結果>

```
d=44.222222
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_02
  悪い例：ファイル名:Question15_02_bad.c
  良い例：ファイル名:Question15_02_good.c

- キャスト演算子を利用して修正すること
- 新たに命令文や変数、算術演算子を追加しないこと
- 変数宣言の型は、変更しないこと

Question15_03

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について (1)型の扱い
・ 情報損失を起こす可能性のあるデータ型への代入(=演算、関数呼び出しの実引数渡し、関数復帰)や演算を行う場合は、問題がないことを確認し、問題がないことを明示するためにキャストを記述する

良い例	悪い例

// 代入の例
short s; // 16 ビット
long w; // 32 ビット
void func() {
s = (short)w;
s = (short)(s + 1);
}
// 演算の例
// int サイズが 16 ビット
unsigned int var1, var2;
var1 = 0x8000;
var2 = 0x8000;
// 判定結果は真
if ((long)var1 + var2 > 0xffff) {
	// 代入の例
	short s; // 16 ビット
	long w; // 32 ビット
	void func() {
	s = w;
	s = s + 1;
	}
	// 演算の例
	// int サイズが 16 ビット
	unsigned int var1, var2;
	var1 = 0x8000;
	var2 = 0x8000;
	// 判定結果は偽
	if (var1 + var2 > 0xffff) {

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{
	int ival=301;
	char cval=ival;
	unsigned short uival=65535;
	ival=uival;

	printf("cval=%d¥n",cval);
	printf("uival=%d¥n",uival);
	printf("ival=%d¥n",ival);

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
cval=45
uival=65535
ival=65535
```

<期待する結果>

```
cval=45
uival=65535
ival=-1
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_03
  悪い例：ファイル名:Question15_03_bad.c
  良い例：ファイル名:Question15_03_good.c

- キャスト演算子を利用して修正すること
- 新たに命令文や変数、算術演算子を追加しないこと
- 変数宣言の型は、変更しないこと
- 変数 cval については、明確な理由をコメントとして記載すること

Question15_04

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について (2) 符号なし変数の扱い
・ unsigned char 型、または unsigned short 型のデータをビット反転（~）、もしくは左シフト（<<）する場合、結果の型に明示的にキャストする
  ➢ 演算によって符号ビットがオンになると、期待した演算結果にならない場合がある

良い例	悪い例

uc = 0x0f;
if((unsigned char)(~uc) >= 0x0f)
	uc = 0x0f;
	//~uc は負の値となるので必ず偽
	if((~uc) >= 0x0f)

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{

	unsigned char uc = 0x0f;
	if((~uc) >= 0x0f){
		printf("0x0f 以上¥n");
	}else{
		printf("0x0f 未満¥n");
	}

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
0x0f 未満
```

<期待する結果>

```
0x0f 以上
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_04
  悪い例：ファイル名:Question15_04_bad.c
  良い例：ファイル名:Question15_04_good.c

- キャスト演算子を利用して修正すること
- 新たに命令文や変数、算術演算子を追加しないこと
- 変数宣言の型や if 文の条件式は、変更しないこと

Question15_05

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について (4)実行時エラーの回避
・ 除算や剰余算の右辺式は、0 でないことを確認してから演算を行う

良い例	悪い例

if (y != 0) {
ans = x/y;
}
	ans = x/y;

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{
        int x,y,ans;
        x=7;
        y=0;
        ans=-1;

    ans = x/y;
      printf("ans の値を表示¥n");
      printf("ans=%d",ans);

        return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果> ※何も表示されない、もしくは、実行時エラーが発生します

```

```

<期待する結果>

```
ans の値を表示
ans=-1
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_05
  悪い例：ファイル名:Question15_05_bad.c
  良い例：ファイル名:Question15_05_good.c

- if 文を利用して、期待する結果になるように修正すること
- 新たに変数、算術演算子を追加しないこと
- 変数宣言の型や初期値は、変更しないこと
- Windows の電卓アプリケーションを動かして、7÷0 をした場合に表示される内容を確認すること

Question15_06

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について　(5)評価順序・演算子の優先順位

- 式中に優先順位の異なる複数の 2 項演算子を含む場合には優先順位を明示するためのかっこをつける[MISRA C:2012 Rule 12.1]

良い例	悪い例

a = (b << 1) + c;
　　または
a = b << (1 + c);
	// 優先順位間違いしている 可能性あり
	a = b << 1 + c;

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>
int main(void)
{
	char a,b,c;
	a=0;
	b=127;
	c=3;

	a = b << 1 + c;
	printf("a=%d",a);

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
a=-16
```

<期待する結果>

```
a=1
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_06
  悪い例：ファイル名:Question15_06_bad.c
  良い例：ファイル名:Question15_06_good.c

- コーディング規約の良い例を参照し、丸かっこ()を利用して、期待する結果になるように修正すること
- 新たに変数、算術演算子、処理を追加しないこと
- 変数宣言の型や初期値は、変更しないこと

Question15_07

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
7 変数・演算について　(5)評価順序・演算子の優先順位

- 論理演算子&&または||の右側のオペランドには、影響があってはならない
  - &&や||演算子の右式は、左式の条件結果により、実行されない場合がある

    [MISRA C:2012　R13.5]

良い例	悪い例

// メモリマップ I/O 用アドレス
volatile int *io_port = ...;
 	
// if 文の条件によらず I/O 処理実施
int io_result = *io_port;
if ((x != 0) && (io_result > 0)) {
　...
}
	// メモリマップ I/O 用アドレス
	volatile int *io_port = ...;
	
	// if 文の条件によって I/O 処理を
	// 行うか否か異なる
	if ((x != 0) && (*io_port > 0)) {
	　...
	}

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>

int func1(){
	printf("func1 が呼ばれました");
	return 2;
}

int main(void)
{
	int x;
	x=2;
	if ((x != 0) || (func1()==0)) {
		printf("if 文は TRUE");
	} else{
		printf("if 文は FALSE");
	}

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
if 文は TRUE
```

<期待する結果>

```
func1 が呼ばれました
if 文は TRUE
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_07
  悪い例：ファイル名:Question15_07_bad.c
  良い例：ファイル名:Question15_07_good.c

- どのように修正すれば、func1 が呼び出されるかを検討して、修正すること
- 変数宣言の型や初期値は、変更しないこと
- 修正プログラムを作成後、インターネットで「C 言語 短絡評価」もしくは「C 言語 ショートサーキット」という単語で検索し、悪い例で func1 が呼び出されなかった理由を確認すること

Question15_08

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
8 比較について (1)禁止事項
・ 区間内の値のうち、特定の実装値のみで判定しない
- 真偽を求める式の中で、真として定義した値と比較してはならない
- C 言語では、真は 0 ではない値でしめされ、1 とは限らない

良い例	悪い例

#define FALSE 0
// func1 は 0 と 1 以外を返す可能性がある
void func2() {
if (func1() != FALSE) {
または
if (func1()) {
	#define TRUE 1
	// func1 は 0 と 1 以外を返す可能性がある
	void func2() {
	if(func1() == TRUE) {

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
#include <stdio.h>
#define TRUE 1

int func1(){
	return 2;
}

void func2() {
	if(func1() == TRUE) {
		printf("func1 の結果は TRUE¥n");
	}else{
		printf("func1 の結果は FALSE¥n");
	}
}

int main(void){
	func2();
	return 0;
}
```

次ページの動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
func1 の結果は FALSE
```

<期待する結果>

```
func1 の結果は TRUE
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_08
  悪い例：ファイル名:Question15_08_bad.c
  良い例：ファイル名:Question15_08_good.c

- コーディング規約の良い例を参考に、プログラムを修正すること
- 修正箇所は、func2 内のみにし、func1 は編集しないこと
- 新たに定数や変数、関数を追加しないこと

Question15_09

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
9 繰り返し・ループの扱い (2)ループカウンタ
・ 浮動小数点型の変数はループカウンタとして使用しない
  - 浮動小数点型はループカウンタとして演算が繰り返されると誤差が累積して意図した結果にならないことがある

良い例	悪い例

void func(void){
  int i;
  for(i=0;i<10;i++){
  }
}
	void func(void){
	  double d;
	  for(d=0.0;i<1.0;d+=0.1){
	  }
	}

9 繰り返し・ループの扱い (3)ループ継続条件
・ ループカウンタの比較に等価演算子(==、!=)は使用しない

良い例	悪い例

void func() {
  int i;
  for (i = 0; i < 9; i += 2) {
  …
	void func() {
	  int i;
	  for (i = 0; i != 9; i += 2) {
	  …

下記のプログラムをコンパイル、実行して結果を確認しましょう

<悪い例:プログラムコード>

```
#include <stdio.h>

int main(void)
{
	double d;
	int i=0;
	for(d=0.0;d!=0.1;d+=0.01){
		printf("%d 回目¥n",i);
		i++;
	}

	return 0;
}
```

次の動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>　※無限ループとなり終了しません。Ctrl+C で強制停止が必要

```
0 回目
1 回目
2 回目
3 回目
4 回目
5 回目
6 回目
7 回目
8 回目
9 回目
10 回目
11 回目
12 回目
   :       -省略-
534 回目
535 回目
536 回目
537 回目
538 回目
^C
```

<期待する結果>

```
0 回目
1 回目
2 回目
3 回目
4 回目
5 回目
6 回目
7 回目
8 回目
9 回目
```

悪い例、期待する結果になるプログラム(良い例)は、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

・　ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
　　ソリューション名:c_training_q15
　　プロジェクト名:Question15_09
　　悪い例：ファイル名:Question15_09_bad.c
　　良い例：ファイル名:Question15_09_good.c

・　コーディング規約の良い例を参考に、プログラムを修正すること

Question15_10

試してみましょう！
研修で利用するコーディング規約にある下記の事項について、実際にプログラムを記述して影響度を確認しましょう。

<コーディング規約の内容>
11.ポインタの扱い (1)禁止事項

- ポインタへの整数の加減算やインクリメント・デクリメントは使用しない
  - ポインタに対する演算はポインタの指している先をわかりにくくする原因となる

良い例	悪い例

#define N 10
char buf[N];
char *p = buf;
int i = 1;
buf[i] = 'a' ;
buf[i+3] = 'c' ;
	#define N 10
	char buf[N];
	char *p = buf;
	*(p+1) = 'a' ;
	p+=2;

下記のプログラムをコンパイル、実行して結果を確認しましょう。
<悪い例:プログラムコード>

```
int main(void) {
	int a[2]={10,20};
	int *pa=a;
	int c=*pa++;
	int d=*pa++;

	int b[2]={10,20};
	int *pb =b;
	int e=(*pb)++;
	int f=(*pb)++;

	if(c==e){
		printf("処理終了後 c:%d と e:%d は同じ値です¥n",c,e);
	}
	if(d==f){
		printf("処理終了後 d:%d と f:%d は同じ値です¥n",d,f);
	}

	return 0;
}
```

次ページの動作結果になることを確認した後、期待する結果になるように良い例に従ってプログラムを書き換えましょう。

<動作結果>

```
処理終了後 c:10 と e:10 は同じ値です
処理終了後 d:20 と f:11 は違う値です
```

<期待する結果>

```
処理終了後 c:10 と e:10 は同じ値です
処理終了後 d:20 と f:20 は同じ値です
```

期待する結果になるプログラムは、下記、<プログラムの条件>に合うプログラムを作成し、実行結果を表示するコードを記述して、コンパイル、実行しましょう。

<プログラムの条件>

- ソリューション名、プロジェクト名、ファイル名は以下の指示に従うこと
  ソリューション名:c_training_q15
  プロジェクト名:Question15_10
  ファイル名:Question15_10.c

- 悪い例のプログラムコードと同様にポインタ変数を利用して、c,d,e,f に値を代入すること
- 必要に応じて、変数を追加すること