ソフト系　C言語実習課題 0
九九の表の作成

v3.22

2023 JTEC m.h　1

これから、専門研修でC言語の実習をします。C言語の課題は、すべて、ステップ by ステップで進めていきます。もし、課題の内容をどのようにするかを理解できていれば、すべて最初からする必要はありません。途中のstepは省略してもかまいません。例えば、九九の表の作成は、step0から始まりますが、Step0とStep1は飛ばして、Step2から始めるということです。 C言語を十分理解している人は、いきなり最後のStep4やStep5からやってもかまいません。自分のスキルに合わせて、できるところから始めてください。

最初の課題は、九九の表の作成で、C言語がほとんど初めてという人が対象です。
for文の2重ループを使って九九の表を作成してみます。

なお、C言語研修の課題は、この課題0のこの九九の作成～課題4 Sレコードの作成まであります。

課題0:九九の表の作成

C言語の基本と1次元配列、乱数の取得、新たに関数を作成するなどを学びます。

課題1: ライフゲームの作成

2次元配列と、戻り値のある関数の作成、テキストファイルの読み込みなどを学びます

課題2:万年カレンダーの作成

カレンダー計算の方法、3次元配列、ビット操作などを学びます。

課題3:dump関数、dumpコマンドの作成

Shift-JISコードの表示、コマンド引数、標準入出力とリダイレクトなどを学びます。

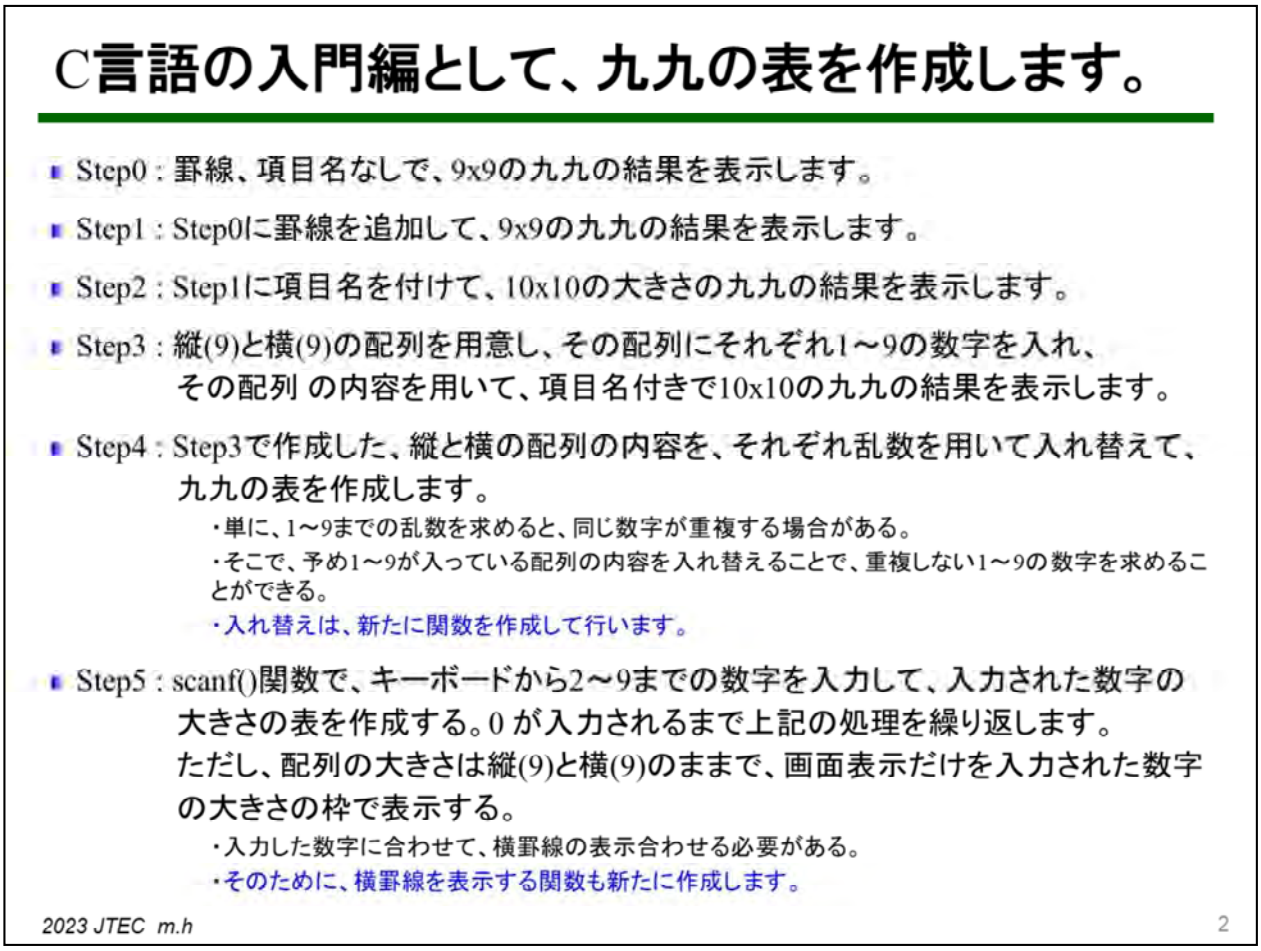
最初にdump関数を作成し、そのあとdump関数を元にdumpコマンドを作成します。

課題4:Sレコードコマンドの作成

ASCII文字列からバイナリ変換、チェックサムなどを学びます。Sレコードは、

「Sレコードからバイナリ」にするコマンド、

「バイナリからSレコード」にする子コマンドの2つを別々にセットで作成します。



Step0 : 罫線、項目名なしで、9x9の九九の結果を表示します。

Step1 : Step0に罫線を追加して、9x9の九九の結果を表示します。

Step2 : Step1に項目名を付けて、10x10の大きさの九九の結果を表示します。

Step3 : 縦(9)と横(9)の配列を用意し、その配列にそれぞれ1～9の数字を入れ、その配列 の内容を用いて、項目名付きで10x10の九九の結果を表示します。

Step4 : Step3で作成した、縦と横の配列の内容を、それぞれ乱数を用いて入れ替えて、九九の表を作成します。

・単に、1～9までの乱数を求めると、同じ数字が重複する場合があります。重複しない1～9までの数字を乱数で作るのは結構大変です。

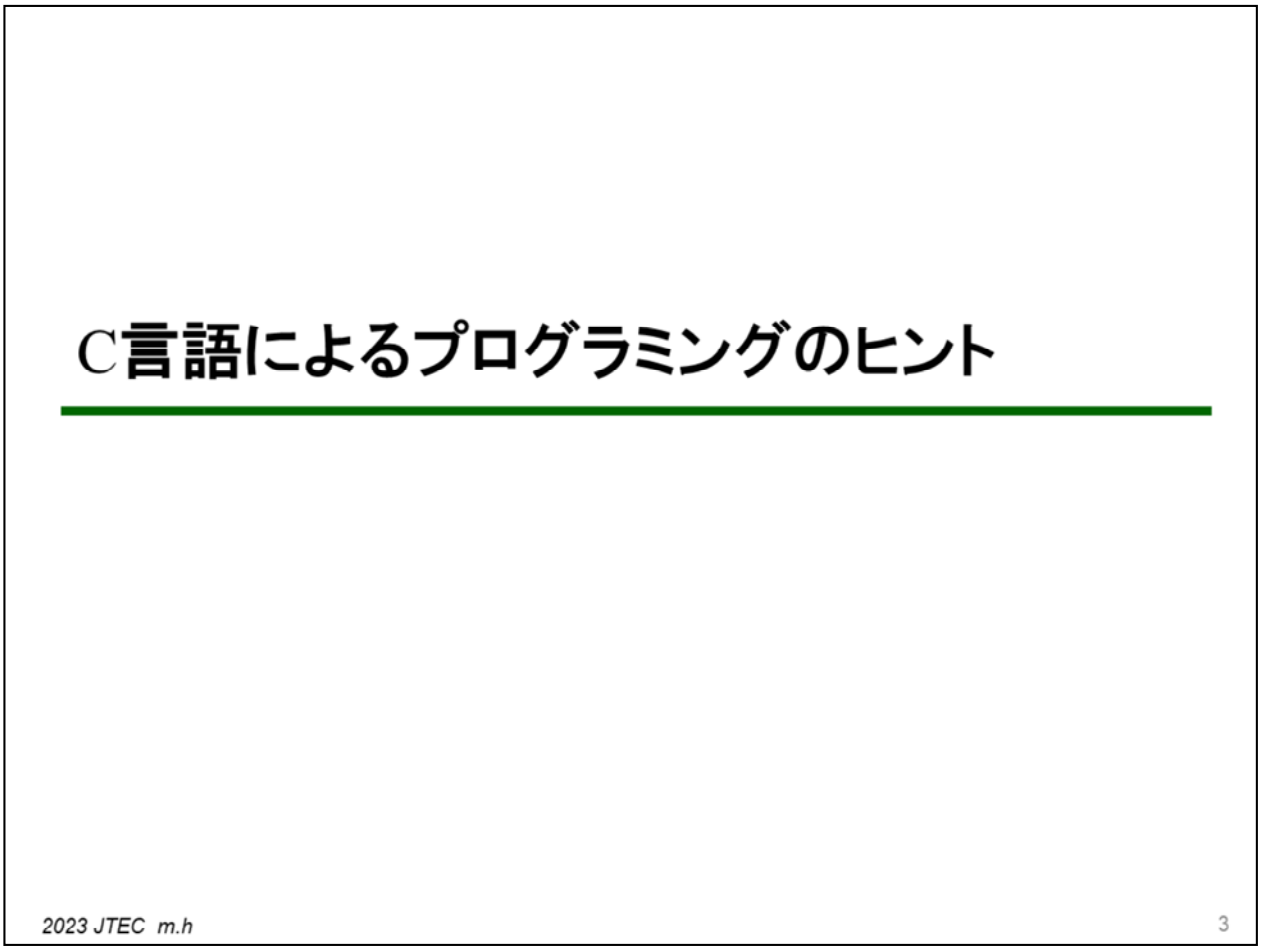
・そこで、予め1～9が入っている配列の内容を入れ替えることで、重複しない1～9の数字を簡単に作成することができます。

・入れ替えは、新たに関数を作成して行います。

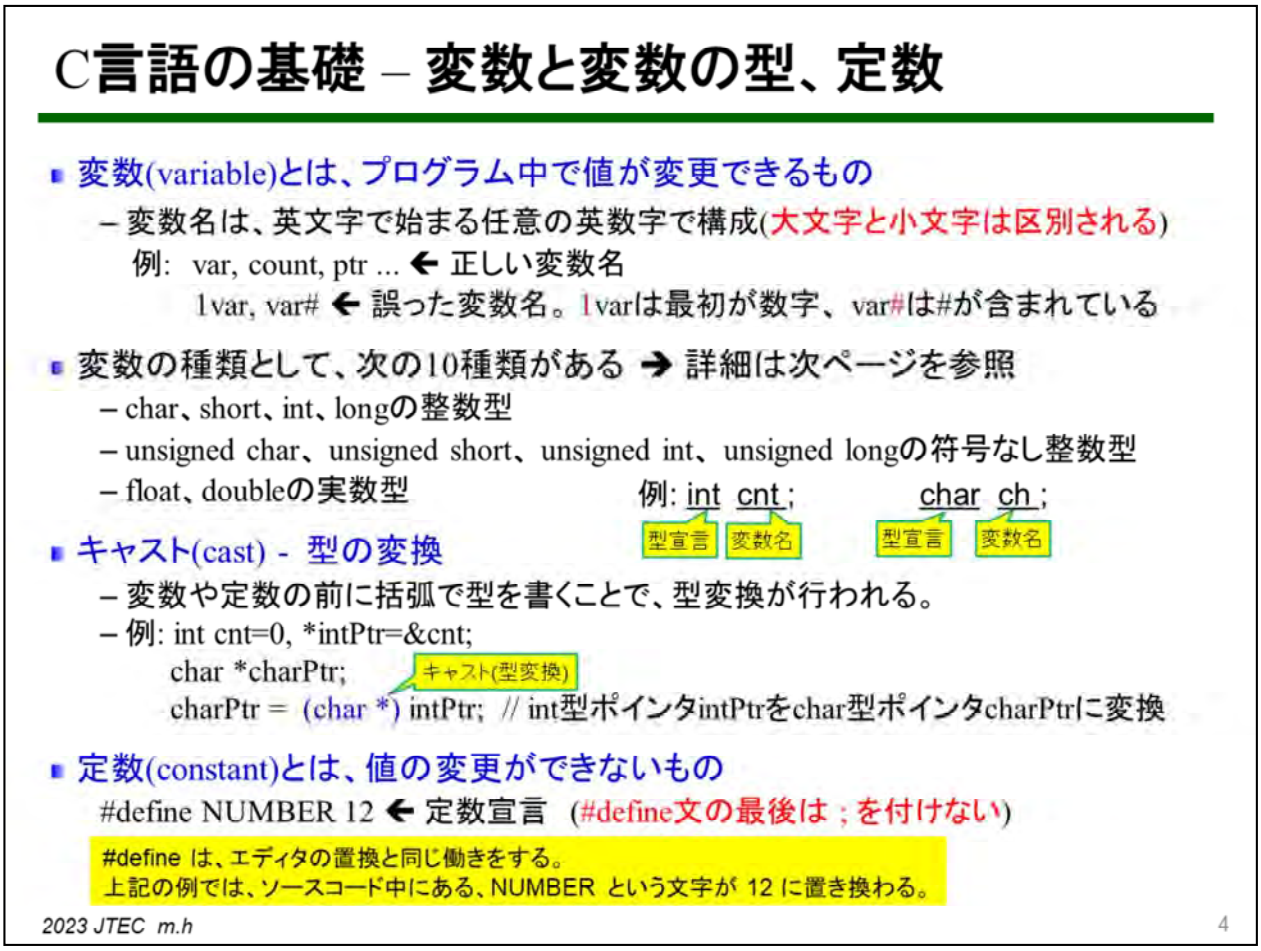
Step5 : scanf()関数で、キーボードから2～9までの数字を入力して、入力された数字の大きさの表を作成すます。0 が入力されるまで上記の処理を繰り返します。

ただし、配列の大きさは縦[9]と横[9]のままで、画面表示だけを入力された数字の大きさの枠で表示します。

・入力された枠数に応じて、横罫線を表示する関数も新たに作成します。



C言語によるプログラミングのヒントとして、「C言語の基礎の基礎」のテキストから一部抜粋して説明をしています。



変数(variable)とは、プログラム中で値が変更できるものです。

変数名は、英文字で始まる任意の英数字で構成されます。そして、大文字と小文字は区別されますので、注意をしてください。

例: var, count, ptr ... は、正しい変数名です。

1var, var# らは 誤った変数名です。1varは最初が数字、var#は#が含まれている

変数の種類として、次の10種類があます。詳細は次々ページを参照してください。

char、short、int、longの整数型

unsigned char、unsigned short、unsigned int、unsigned longの符号なし整数型

float、doubleの実数型

そして、変数定義は、型の後に変数名を書きます。変数名は、型の後に、,(カンマ)で区切って複数書くことができます。ただ、型が同じといって、用途の違う変数を並べのるは、プログラム的には、エラーにはなりませんが奨励しません。可読性を高めるために、1変数を1行にして何の変数からのコメントを入れるようにしてください。

キャスト(cast) - 型の変換

変数や定数の前に括弧で型を書くことで、強制的に型変換が行われます。

例: int cnt=0, *intPtr=&cnt;

```
char *charPtr;
charPtr =  (char *) intPtr;  // int型ポインタintPtrをchar型ポインタcharPtrに変換
```

このキャストを行うとで、整数型(32ビット4バイト)り変数を1バイト単位でアクセスできるようになります。

以前のC言語仕様では、型が違った場合、警告で済んでいましたが、最近ではエラーになることがあります。

定数(constant)とは、値の変更ができないものです。

#define NUMBER 12 □ 定数宣言 (#define文の最後は ; を付けない)

C言語で用いる、#define は、プリプロセッサで、エディタの置換と同じ働きをします。ここでも、C言語の初級者が間違うのが、#defineの最後に ; を付けてしまうことです。#define文には、; は付かないと覚えてください。

名前の付け方のルールついて

- 英文字の関数名や変数名には、空白は使えない
  - そのため、いくつかのルール(作法)を定め可読性を良くしている
- キャメルケース
  - ローワーキャメルケース (LCC：Lower Camel Case)
    - 先頭の1文字を小文字で始める
    - 例: getStatus, printStatus, fileName
    - JavaやJavaScriptで使われている
  - アッパーキャメルケース (UCC：Upper Camel Case)
    - 先頭の1文字を大文字で始める
    - 例: GetStatus, PrintStatus, FileName
    - WindowsのAPI、C#、VBAなどで使われている
- スネークケース
  - 英単語を「_」(アンダースコア)でつなげる表記
    - すべて小文字か大文字表記のときに見やすくなる
    - 例: get_status, print_status, file_name, W_LIST、MAX_SIZE
- C言語では、いずれ付け方でもかまわないが、ソースコード全体で統一する
  - マイルールを決めると良いが、客先でのコーデングルールを優先すること

キャメル(Camel)は、「らくだ」のこと

定数は、すべて大文字のスネークケースにする

2021 JTEC m.h

変数名や関数名は、その役割が分かるよう命名にします。
C言語では、英文字の関数名や変数名には、空白は使えないません。
そのため、いくつかのルール(作法)を定め可読性を良くしています。
【キャメルケース】
ローワーキャメルケース (LCC：Lower Camel Case)
先頭の1文字を小文字で始めます。
例: getStatus, printStatus, fileName
JavaやJavaScriptで使われています。
【アッパーキャメルケース】(UCC：Upper Camel Case)
先頭の1文字を大文字で始めます。
例: GetStatus. PrintStatus, FileName
WindowsのAPI、C#、VBAなどで使われています。
【スネークケース】
英単語を「__」(アンダースコア)でつなげる表記です。
すべて小文字か大文字表記のときに見やすくなります。
例: get_status, print_status, file_name, W_LIST、MAX_SIZE
C言語では、いずれ付け方でもかまわないが、ソースコード全体で統一します。

なお、定数はすべて大文字にします。長い文字の場合はスネークケースにします。

マイルールを決めると良いが、客先でのコーデングルールを優先してください。

C言語で扱う変数の種類

分類	変数の型	ビット長	バイト数[*1)]	記憶できる数値の範囲	備考
整数型	char	8 bit	1	−128～+127	文字型
	short	16 bit	2	−32,768～+32,767	短整数型
	int	32 bit	4	−2,147,483,648～+2,147,483,647	処理系依存[*2)]
		16 bit	2	−32,768～+32,767	
	long	32 bit	4	−2,147,483,648～+2,147,483,647	長整数型
	unsigned char	8 bit	1	0～+255	符号なしchar型
	unsigned short	16 bit	2	0～+65,535	符号なしshort型
	unsigned int	32 bit	4	0～4,294,967,295	符号なしint型[*2)]
		16 bit	2	0～+65,535	
	unsigned long	32 bit	4	0～4,294,967,295	符号なしlong型
実数型	float	32 bit	4	-2^{+127}～-2^{-128}, $+2^{-128}$～$+2^{+127}$	単精度実数型
	double	64 bit	8	-2^{+1023}～-2^{-1024}, $+2^{-1024}$～$+2^{+1023}$	倍精度実数型

*1) バイト数は、sizeof(型)、sizeof(変数)で返される値 (sizeof()は型や変数のメモリサイズを返す演算子)。
ただし、引数で受け取った変数は、ポインタのサイズとなるためsizeof() 使用はできないため注意が必要。

*2) int型は、現在では32ビットが一般的だが、8ビットや16ビットのCPUで使用する場合、16ビットとして扱われることもある。正確な変数のサイズは、sizeof() 演算子で確認したほうが良い。

2023 JTEC m.h 6

この表は。C言語で使う変数の型の一覧です。

C言語は、他の言語に比べると型の種類は少ないです。

例えば、C言語には、C++ にある、trueかfalseかの2値を持つbool型はありません。char型で代用します。

専門研修の課題で多く使用するのは、int型とchar型です。そして、char型は、符号なしのunsinged charも使います。その他の型については、使用する場面が登場したとき使ってください。

ただ、変数の型とサイズは覚えておきましょう。

この表で、int型は、処理系依存となっています。これは、コンピュータが8ビット、16ビット、32ビットと進化の過程で、int型のサイズは変化してきました。昔、8ビットや16ビットのCPUの場合は、int型は16ビット(2バイト)でした。

しかし、現在、PCで使われているPCは64ビットとなり、int型は32ビット(4バイト)が一般的になりました。

メモリが十分に使えるPCの世界では、何でもint型で処理しますが、メモリが十分に使えない8ビットや16ビットCPUを使う組込み系ではchar型やshort型も良く使います。

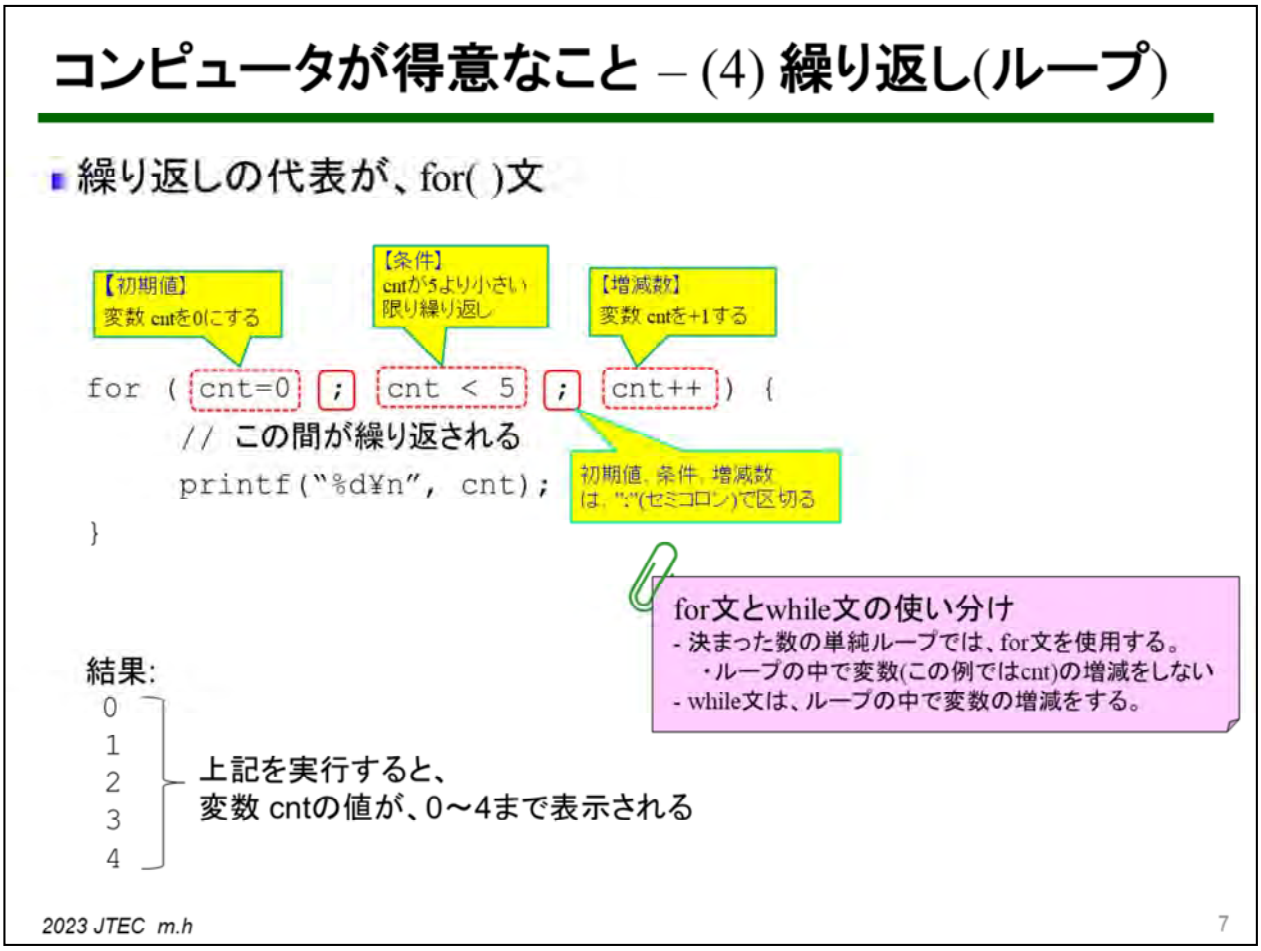
これは、少しでも、使用するメモリ量を減らすためです。

int型が32ビットの場合、char型に比べて4倍メモリを消費します。単純に1つの変数なら大したことはないですが、1000個配列にすると大分違ってきます。

自分が使っている環境で、int型のサイズを知るには、sizeofを使うと分かります。

Visual Studioで、printf (sizeof(int)); を実行すると、4 と表示されます。

ちなみに、VBAのintに相当するIntegerは、16ビットです。



コンピュータが得意なこととして、繰り返しがあります。人間なら飽きてしまうことを何十回、何万回と繰り返してくれいます。

その繰り返しの代表がfor文になります。 for文も、ほとんどすべて言語にあります。

for文は、( )の中に、

for (変数名 = 初期値 ； 条件式 ； 変数の増減値)

と書きます、

この例では、

```
for ( cnt=0  ;  cnt < 5  ;  cnt++ ) {
      // この間が繰り返される
      printf("%d¥n", cnt);
}
```

となっていて、この場合、

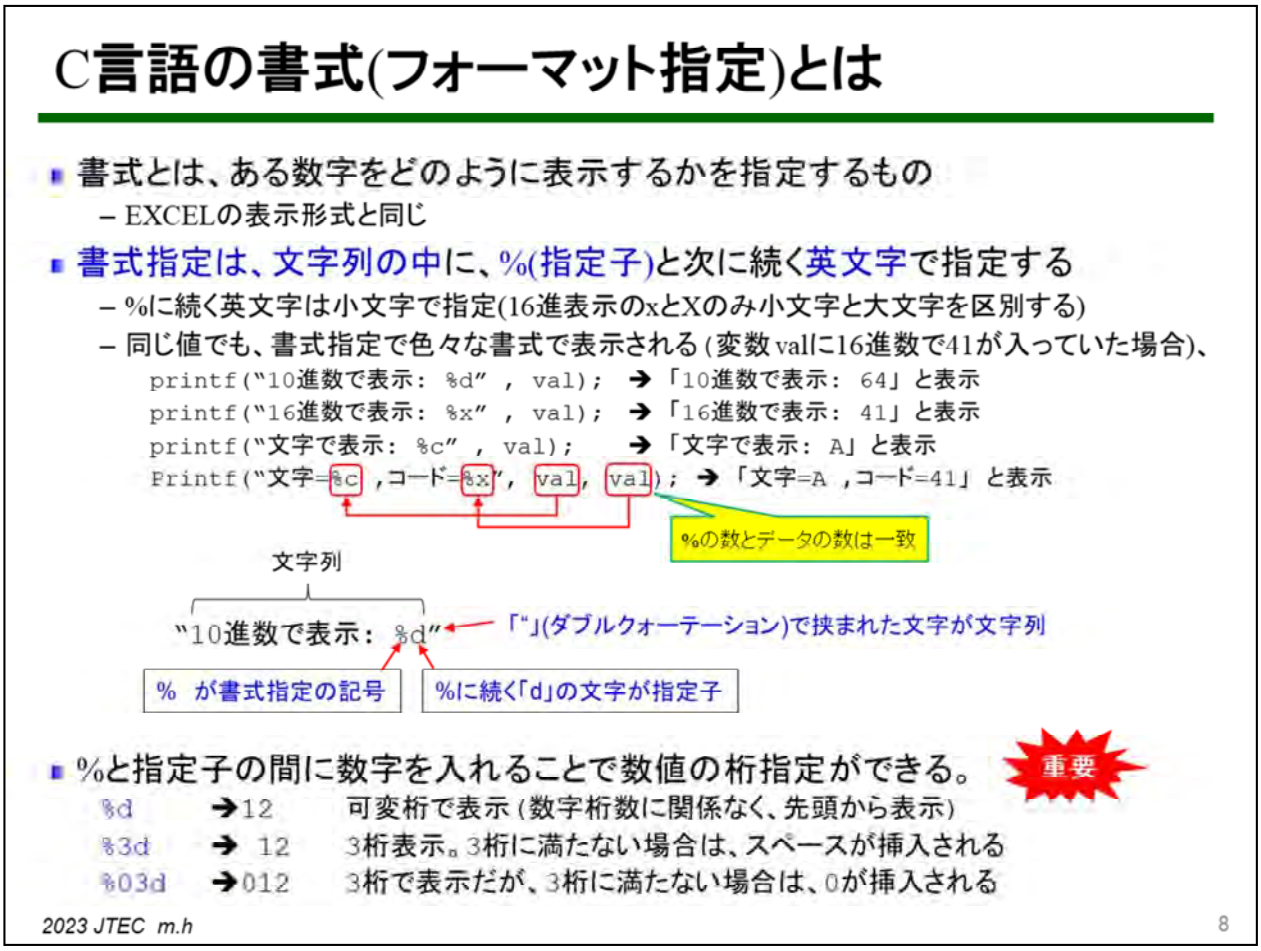
cnt = 0 が初期値、cnt < 5 が条件式、cnt++ が増減値となります。

この例では、cntが0から始まり、5より小さい限りループをすることになります。

結果的には、

```
printf("%d¥n", cnt);
```

が5回繰り返されます。



続いて、C言語における書式指定(フォーマット指定)です。コンピュータが扱うのは、数値であって、文字とか文字列の区別はありません。これは、人間が数値に意味を持たして使っているだけです。

書式指定とは、その数値をどのように表現するかを指定するものです。

この書式指定は、EXCELの表示形式と同じ意味です。例えば、EXCELのセルに、2021/4/1 と入力してみてください。そして、セルの書式設定で、表示形式を「数値」にすると、44282という数字になります。この数字は、1990/1/1から2021/4/1までの累積日数です。次に、セルの書式設定で、表示形式を「2012年3月14日」を選択すると、2021年4月1日と表示されます。あるいは、英語の表記もできます。このように、セルの中の44282という数字が書式設定で色々な表現ができるわけです。

C言語の書式指定もまったく同じです。

C言語でで、文字列の中に、%(指定子)と次に続く英文字で指定します。%に続く英文字は小文字で指定(16進表示のxとXのみ小文字と大文字を区別します)。

例えば、変数 valに16進数で41が入っていた場合、書式指定で下記のような表示になります。

1) printf("10進数で表示: %d" , val); 「10進数で表示: 64」と表示

2) printf("16進数で表示: %x" , val); 「16進数で表示: 41」と表示

3) printf("文字で表示: %c" , val); 「文字で表示: A」と表示

4) printf("文字=%c ,コード=%x", val, val); 「文字=A ,コード=41」と表示

注意することとして、%の数と対応する値の個数は一致している必要があります。上記の4)の例では、%が2つあり、対応するvalという値(変数)が2つあります。

数字を表示するとき、表示桁を指定したい場合があります。そのときは、%と指定子との間数字を入れます。

%d ➔12 可変桁で表示(数字桁数に関係なく、先頭から表示)

%3d ➔ 12 3桁表示。3桁に満たない場合は、スペースが挿入される ← 九九の表で使います

%03d ➔012 3桁で表示だが、3桁に満たない場合は、0が挿入される

この中で、%3dなどの指定は、C言語実習で、研修課題の、九九の表やカレンダーで使います。

C言語の書式一覧(フォーマット指定)

- 下表の中で、最初の4種類は、専門研修のC言語実習で使用します。

%d (Dicimail:10進数) / %x (heXadecimal : 16進数) / %s (String: 文字列) / %c (Character: 文字)

指定子	対応する型	説明	使用例
%c	char	1文字を出力する	"%c"
%s	char *	文字列を出力する	"%8s", "%-10s"
%d	int, short	整数を10進で出力する	"%-2d","%03d"
%x	int, short ,unsigned int, unsigned short	整数を16進で出力する。%Xは、A～Fが大文字	"%04x"
%p	int, short ,unsigned int, unsigned short	ポインタとして16進数8桁で出力する(%08X と同じ)	"%p"
%u	unsigned int, unsigned short	符号なし整数を10進で出力する	"%2u","%02u"
%o	int, short ,unsigned int, unsigned short	整数を8進数で出力する	"%06o","%03o"
%f	float	実数を出力する	"%5.2f"
%e	float	実数を指数表示で出力する	"%5.3e"
%g	float	実数を最適な形式で出力する	"%g"
%ld	long	倍精度整数を10進で出力する	"%-10ld"
%lu	unsigned long	符号なし倍精度整数を10進で出力する	"%10lu"
%lo	long, unsigned long	倍精度整数を8進で出力する	"%12lo"
%lx	long, unsigned long	倍精度整数を16進で出力する	"%08lx"
%lf	double	倍精度実数を出力する	"%8.3lf"

専門研修で使用

2023 JTEC m.h 9

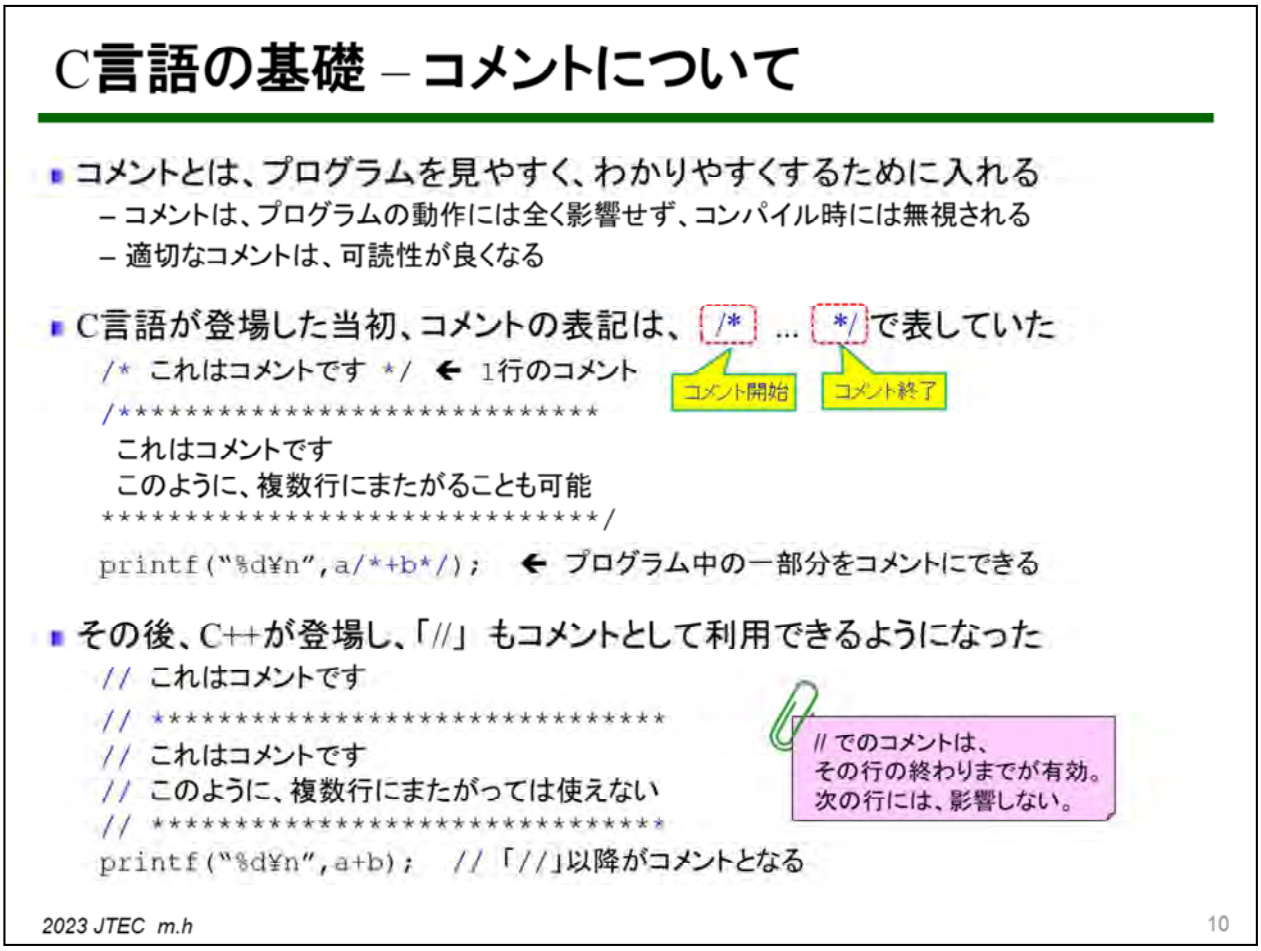
この表は、C言語の書式指定の一覧です。これは、覚えるしかありません。
特に、表の最初の4種類は、専門研修のC言語実習で使用します。
ただ、こうした記号を覚えるには、意味を持たせると忘れにくいです。
特に使用頻度の高い、次の4つは、略語の意味を覚えると良いでしょう。
%d (Dicimail:10進数)
%x (heXadecimal : 16進数)
%s (String: 文字列)
%c (Character: 文字

指定子は、%X を除いてすべて小文字になります。大文字で書くとエラーになります。
%xのみ、大文字があります。%xは、16進数での出力になりますが、%x と小文字にすると、16進数のA～Fが小文字で、%X にすると16進数のA～Fが大文字で出力されます。

また、16進数表示で、%pがあります。これは、ポインタとして出力する書式で、%08X と同等になります。

```
Int num = 0x123abc;
printf("%08x¥n",num);      ➔ 00123abc
printf("%08X¥n",num);      ➔ 00123ABC
printf("%p¥n",num);        ➔ 00123ABC
```

%pで表示される16進数は大文字になります。



まず、最初はコメントの書き方です。どのような言語でもコメントは書けます。コメントは、プログラムには影響しませんが、ソースコードの可読性(読みやすさ)を向上させます。

C言語が登場したとき、コメントの書き方は、/* で始まり */ までがコメントとして扱われました。この、/* ～ */ は、複数行に渡ったり、あるいは1行中の一部でも適応されます。

しかし、* を入力するのに、Shiftキーを押さなければならず、ソースコードの入力時のタイプング速度を低下させていました。

それで、C++が登場したとき、// もコメントとして使えるようになりました。現在では、この // コメントがC言語でも使えるようになっています。

したがって、最近で、入力の簡単な // コメントが多く使われていますが、/* ～*/も、時には と便利にので、今でも使われています。

/* ～ */ は、複数行に渡って作用するため、一時的に、関数全体のコメントにすることができます。さらに、/* ～ */ コメントは、1行中の一部分だけをコメントにすることができます。

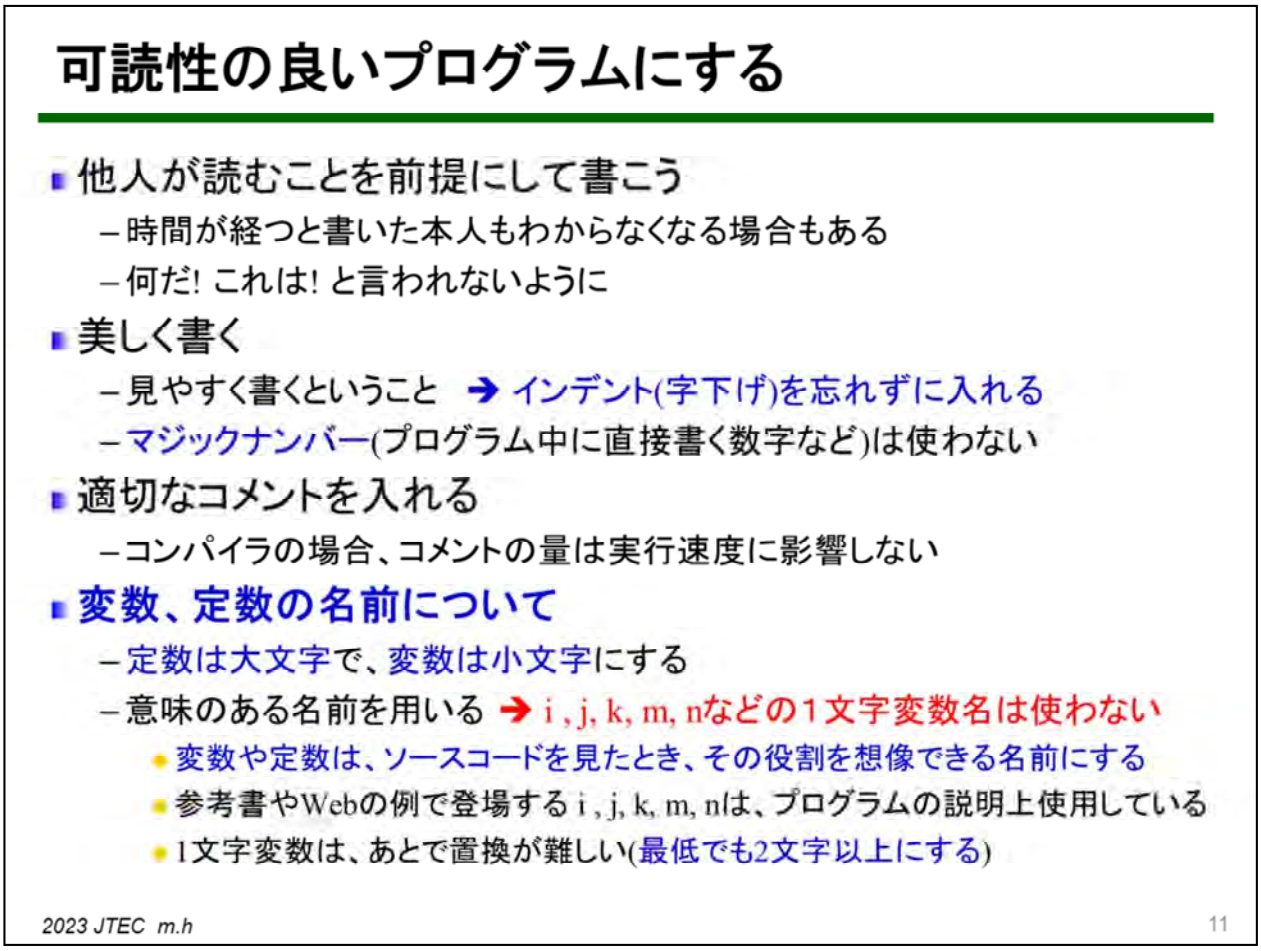
例えば、 `printf("%d¥n",a+b);` を `printf("%d¥n",a/*+b*/);` とすることで、+b だけをコメント化できます。この場合、 `printf("%d¥n",a);` と同等になります。

一方、// コメントの有効範囲は、// からその行の最後までとなります。複数行に渡っては書けません。

「一般的なCコンパイラの流れ」のところでも説明をしましたが、C言語のコメントは、コンパイル前のプリプロセッサですべて取り除かれます。

また、コメントの書き方には、インラインコメントと、プログラム行の右側に書く、行コメントがあります。関数や処理の先頭には、インラインコメンとで、処理の途中は、行コメンで補足する感じです。

コメントの書き方は、色々に作法があり、規約となっている場合もあります。最終的には、客先ルールに従ってください。 大事なことは、可読性を考慮して、コメントの書き方に一貫性を持たせることです。



プログラムを書くとき、単に動作すれば良いと言うものではありません。一度、作成したプログラムは、のちのち変更が容易だとか、再利用ができるとかと言ったことも考慮して作成することが重要です。

そのためには、可読性の良いプログラムにする必要があります。可読性ということは、第三者が見ても、容易に内容が理解できるということです。

「他人が読むことを前提にして書こう」 よくあることですが、実際にプログラムを書いた本人でさえ、時間が経つと、どうしてこうやって書いたのかなどを忘れてしまうこともあります。客先に行って、「何だ! これは! 」と言われないようなプログラムを書いてください。

「美しく書く」 美しく書くということは、見やすく書くということです。上級者が書いたプログラムは、とても読みやすく美しく感じます。見やすいプログラムとは、ブロック毎にインデント(字下げ)がされています。Visual Studioでは、自動的にインデントが挿入されたりしますが、そうでない開発環境もあります。インデントは、バグの発見も容易にします。C言語では、1行の間違いで、ものすごい数のエラーになることがあります。その場合、大体、一番最初のエラーか、インデントに問題があります。

次に、プログラム中、将来変更になるかも知れない数字を直接書かないようにしてください。プログラム中に直接数字を書くことを「マジックナンバー」と言って、作法的に控えるようにしてください。

九九の表の作成では、「9」という数字が何個が登場します。もし、九九の表を10x10に仕様変更する要求があった場合、何カ所もソースを変更しなければなりません。それを、例えば、

```
#define MAXSIZE 9
```

このように最大数を定義しておけば、MAXSIZEの9を10に変更するだけですみます。変更箇所が少ないということは、それだけ、バグの発生する確率が下がります。

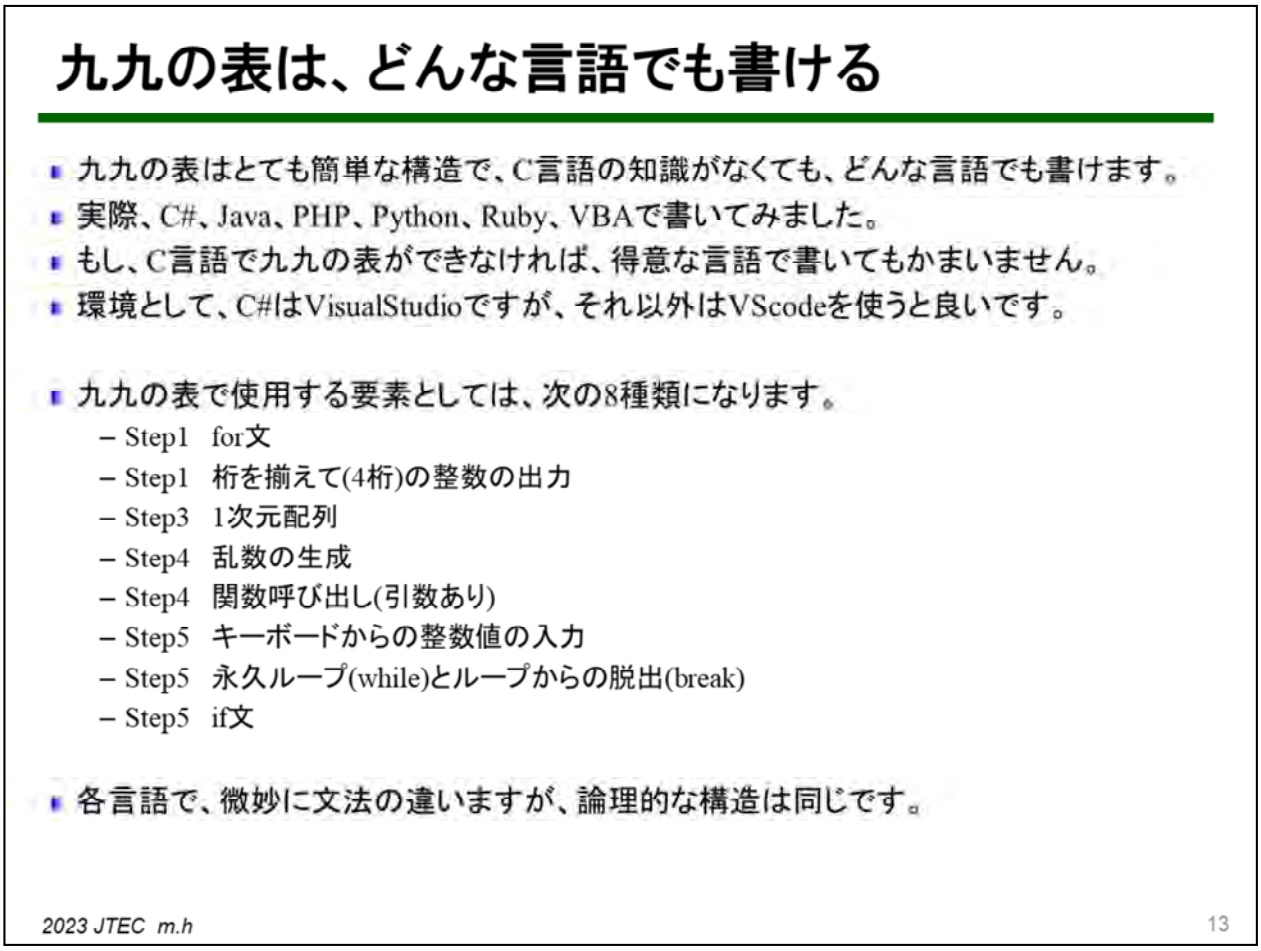
「適切なコメントを入れる」 適切なコメントは、可読性が高まります。コメントは、ソースの行間に入れるコメントと、1行(文)の終わりに入れるコメントの2種類があります。あるまとまった処理をする前には行間コメントを、処理の途中は、必要に応じて、文の最後にコメントを付けます。

「変数、定数の名前について」 定数名変数名は、定数は大文字で、変数は小文字にするのが原則です。これも作法ですが、上級者は、このルールで大文字は定数、小文字は変数と思ってソースを読んでいきます。

また、名前は意味のあるものにしてください。九九でもfor文の二重ループが登場しますが、その中の変数を「i」とか「j」とかにしないでください。縦と横が分かる変数名にします。

あと、名前は基本的には、3文字以上にするのが好ましいです。

それでは、実際の課題として九九の表を作成してみましょう。

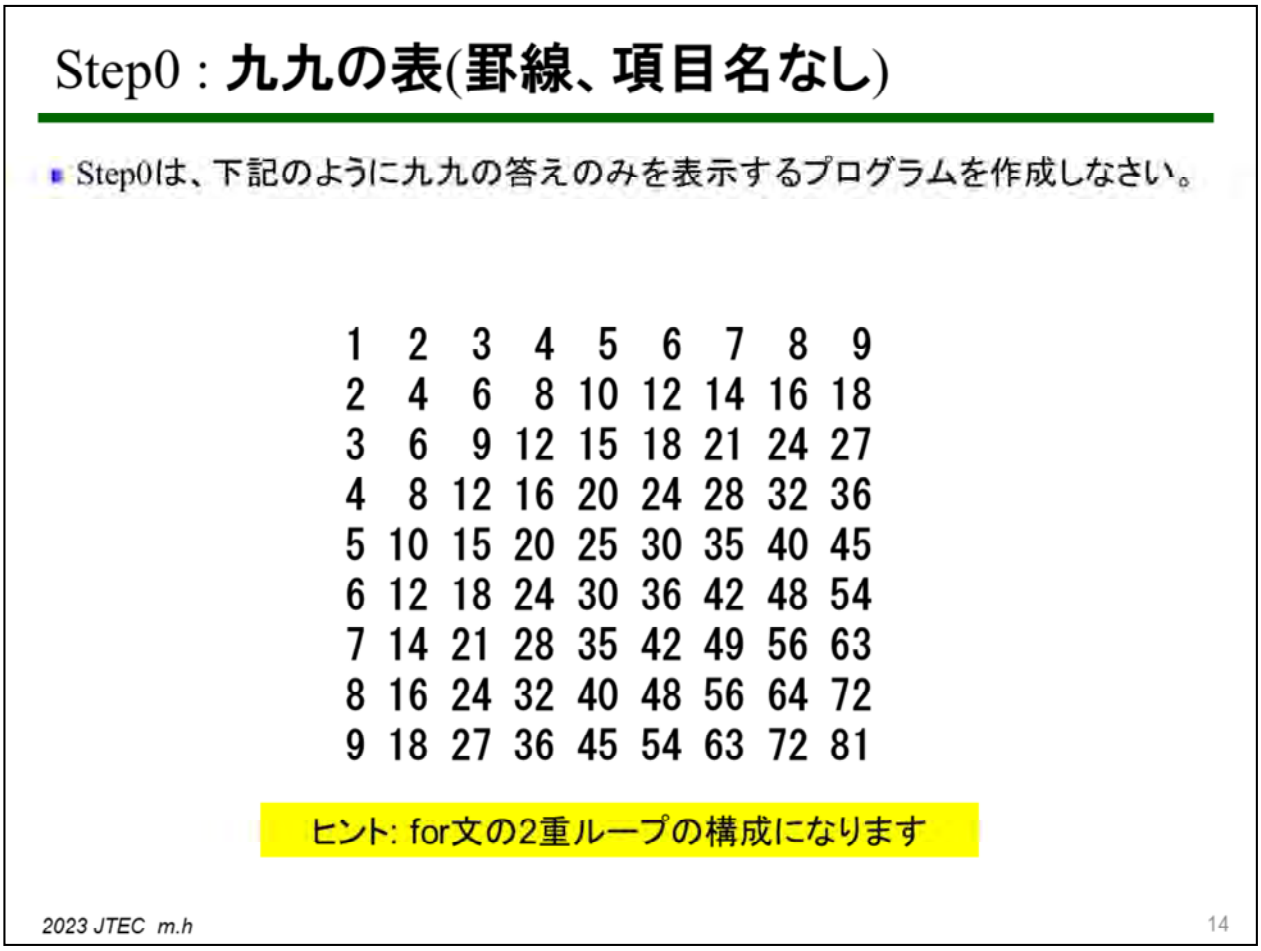


課題0の九九の表は、基本的にfor文の2重ループ構造となります。
そのため、C言語の知識がなくても、どんな言語でも書けます。
実際、C#、Java、PHP、Python、Ruby、VBAで書いてみました。
もし、C言語で九九の表ができなければ、自分の知っている言語で書いてもかまいません。
環境として、C#はVisualStudioですが、それ以外はVScodeを使うと良いです。

九九の表で使用する要素としては、次の8種類になります。
それぞれの違いをネットで調べれはできます。

- Step1 - for文
- Step1 - 桁を揃えて(4桁)の整数の出力
- Step3 - 1次元配列
- Step4 - 乱数の生成
- Step4 - 関数呼び出し(引数あり)
- Step5 - キーボードからの整数値の入力
- Step5 - 永久ループ(while)とループからの脱出(break)
- Step5 - if文

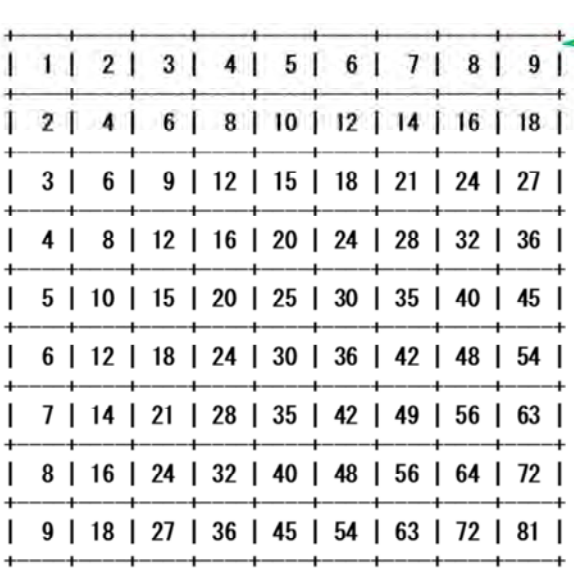
各言語で、微妙に文法の違いますが、論理的な構造は同じです。



Step0として、このような表示をさせてください。
ヒントにもあるように、for文の二重ループにします。つまりfor文の中に。もうひとつfor文が入ります。
使用する関数は、
for()
printf()
の2つだけでできます。

Step1：九九の表に罫線を追加する

- Step1は、下記のように、Step0に罫線を追加して、九九の答えのみを表示するプログラムにしなさい。



2023 JTEC m.h 15

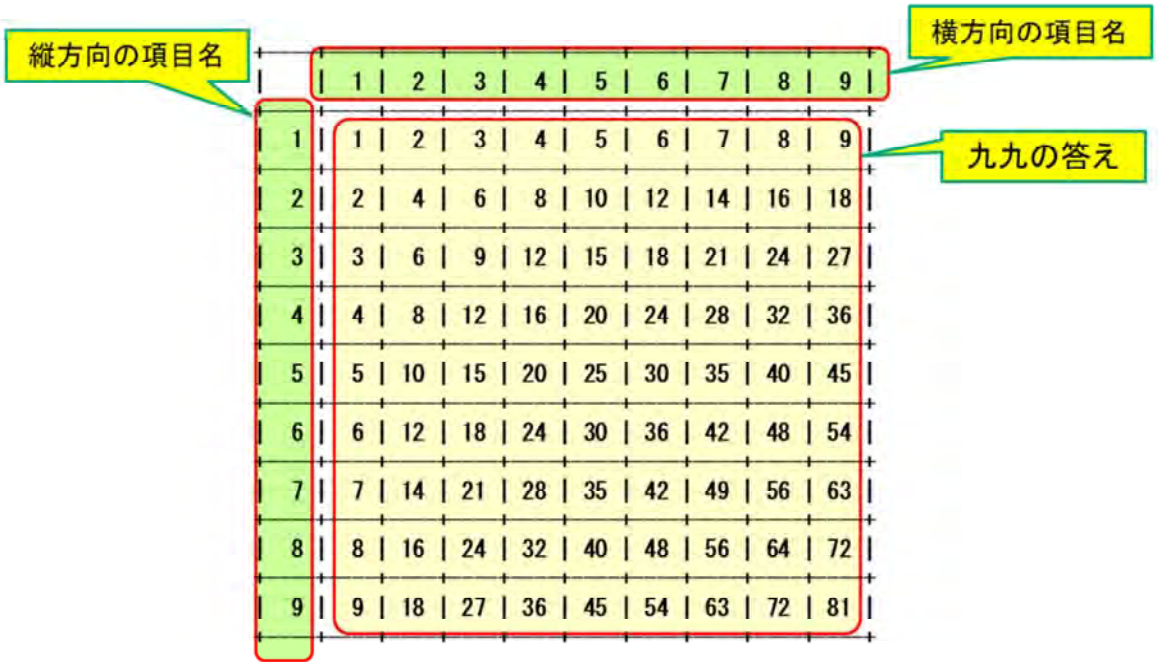
Step1は、Step0で作成したプログラムに、罫線を表示するようにします。
罫線は、「+」、「-」、「|」の3種類の半角文字を組み合わせて表示します。
使用する関数は、Step0と同じです。
printf( )で数字を出力するところで、「|」もいっしょに表示します。

もうひとつヒントとして、横罫線は。下記のようにしします。
printf(“+----+----+----+----+----+----+----+----+----+”); ← 9組

この横罫線の表示は、2回登場します。

Step2：九九の表に項目名(index)を追加する

- Step2は、Step1に対して、下記のように、縦と横に項目名を表示するようにしなさい。



Step2は、Step1に対して、縦と横の項目(index)を表示するようにします。
ヒントを箇条書きにします。
(1) 最初の罫線の表示
`printf(“+----+----+----+----+----+----+----+----+----+”);` ← 10組
で、一番上の横罫線を表示する。この時、罫線の数は縦の項目が入るので1つ増えます。

(2) 横の項目の表示
最初に、”|　“を出力したあと、横の項目をfor文で表示します。

(3) 縦のfor文開始

(4) 縦項目を表示したあと、九九答えを表示
Step1で作成したforの2重ループで、横のループのfor文が始まる前に、「| 」と縦の項目を表示します。

(5) 横のfor文開始
縦のfor文の変数と横のfor文の変数から計算して、九九の答えを表示する

Step3：項目名の部分を配列にする

- Step3は、Step2に対して、縦用と横用に9要素の配列を用意し、その中に、1～9までの数字を入れて、その数字を用いて、Step2の表示を行うようにしなさい。

2023 JTEC m.h 17

Step3では、Step2で作成した骨格を利用して、項目を1次元配列にして、九九の答えも配列の中のデータを使うようにします。

1次元配列は、縦用、横用の2つ定義します。

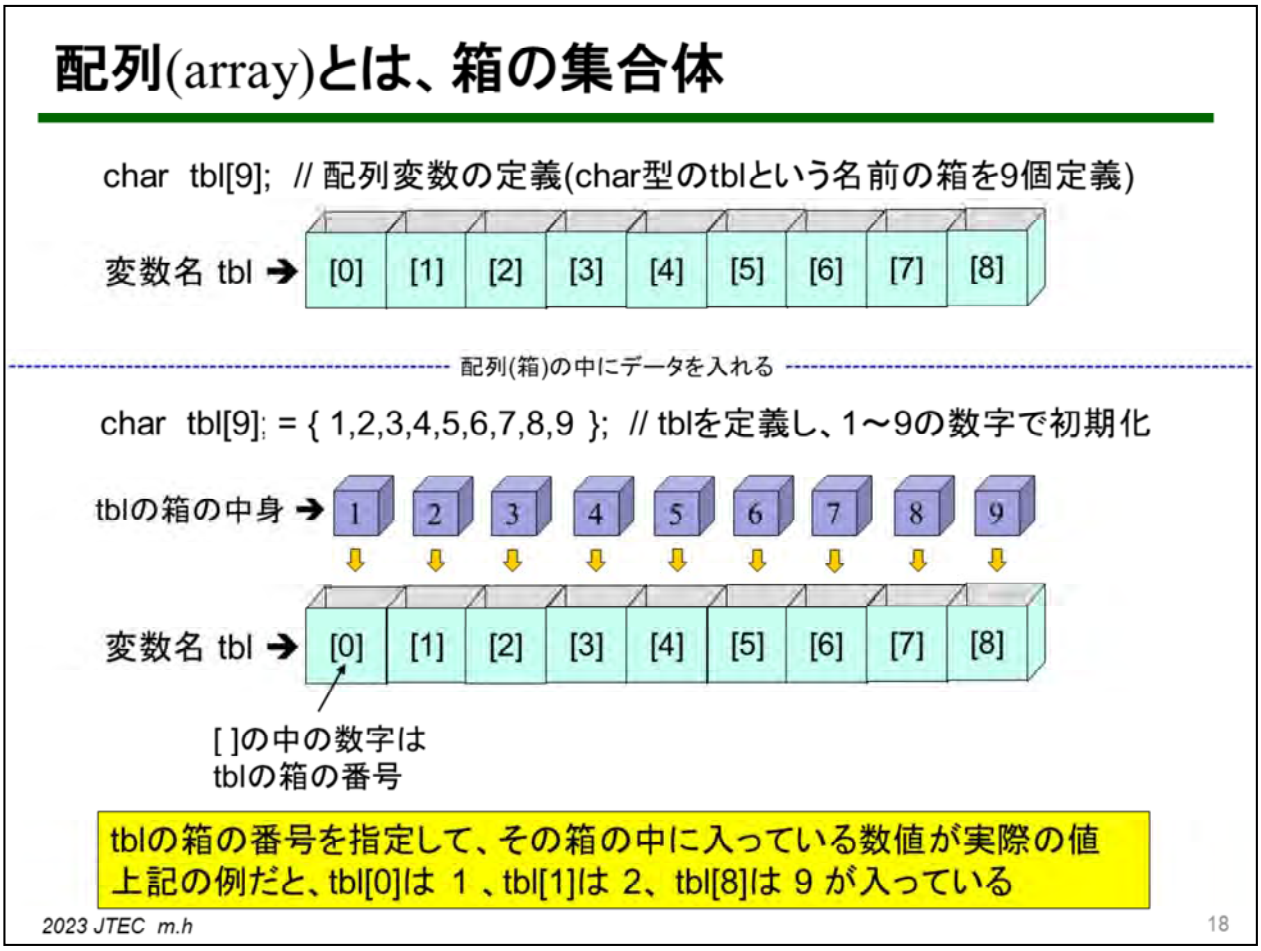
Step2までは、項目も九九の答えも、for文の変数を使っていたと思いますが、Step3では、配列の中の要素を使います。

なお、ここでは、説明上、row_tbl[9]、col_tbl[9]というように、9というマジックナンバーを使っていますが、実際は、配列の大きさは定数で定義してください。

#define MAXSIZE 9

row_tbl[MAXSIZE]、col_tbl[MAXSIZE]

配列については、次のページを参考にしてください。



配列(array)とは、箱の集合体と思ってください。つまり、同じ型のデータの入れ物(箱)を一列に並べたものです。箱の数は、変数の後の[ ]の中に数字を入れて指定します。

この図で、char tbl[9]とは、char型のtblという名前の変数(箱)を9個並べるということになります。

そして、箱には0番から順番に番号が付いています。

char tbl[9]の例では、

tbl[0]、tbl[1]、tbl[2]、tbl[3]、tbl[4]、tbl[5]、tbl[6]、tbl[7]、tbl[8]

というように、tbl0]~tbl[8]までの9個の箱が用意されます。

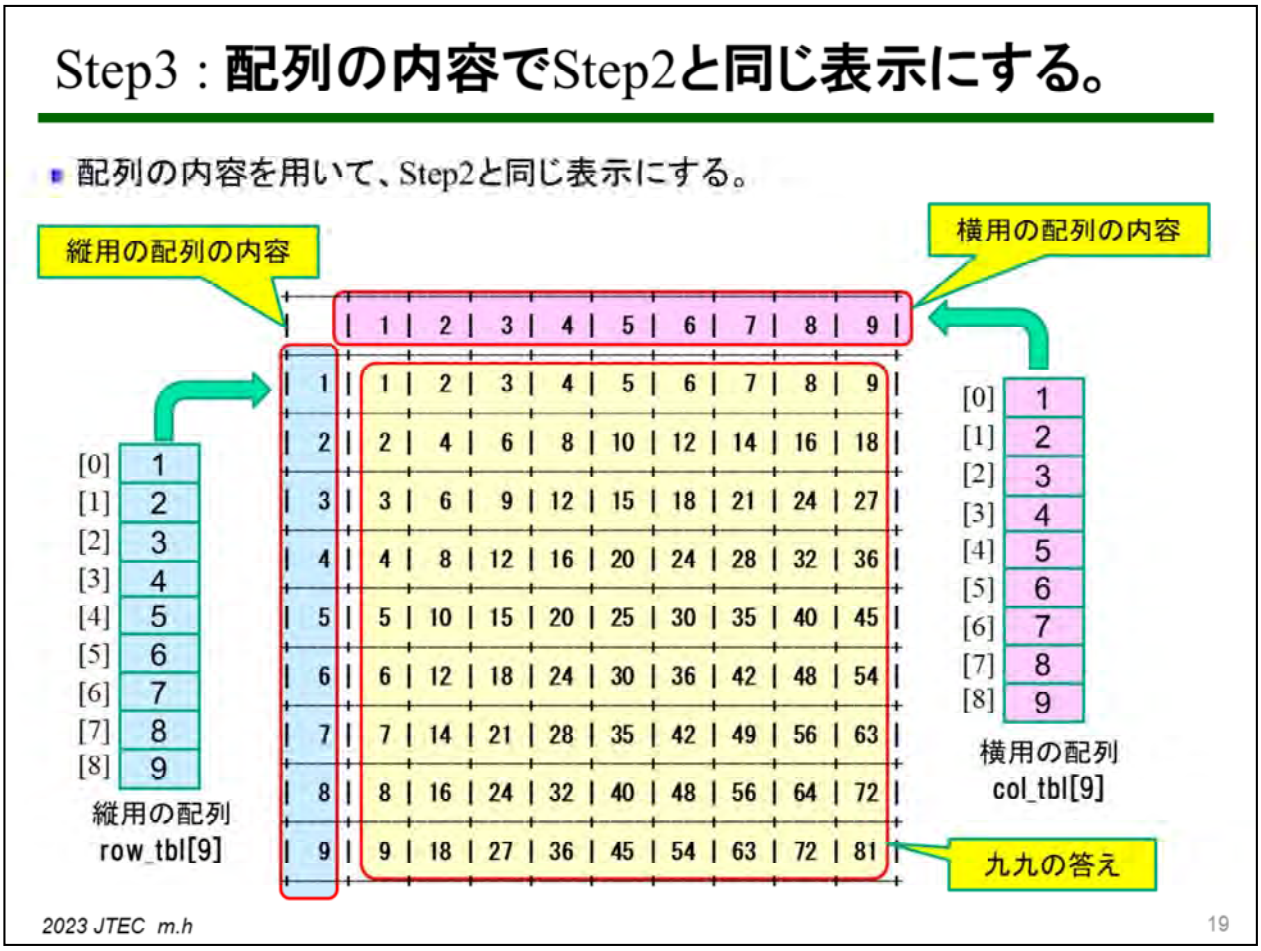
この例では、配列変数の定義だけですが、定義時に、初期値としてデータを入れることもできます。

char tbl[9] = { 1,2,3,4,5,6,7,8,9 };

と書くと、9個の箱の中に、初期値として1～9までの数値が入ります。

変数名tbl[数字]として、[ ]の中の数字が箱の番号になります。その箱の中に入っている数値が実際の値になります。

この例では、0番の箱には1が、1番の箱には2が、そして8番の箱に9が入ることになります。



Step2までは、項目も九九の答えも、for文の変数で表示をしていましたが、Step3では、定義した配列の内容を使って表示します。

そのため、Step3では、for文の変数は配列の要素[箱番号]を指定することになります。

したがって、for文は、1からでなく、0～8までの9回のループになります。

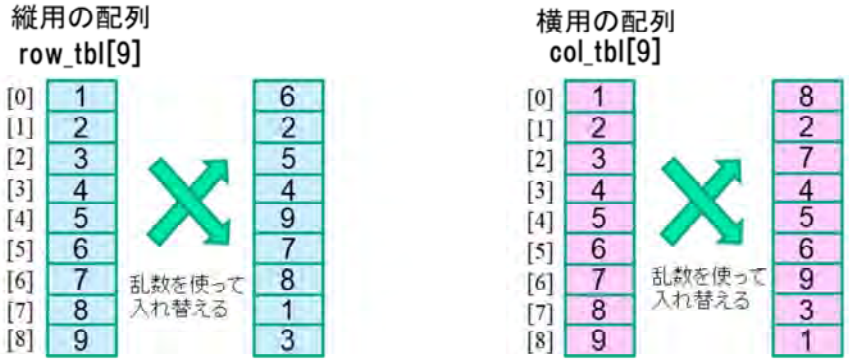
つまり、for文の変数はループ回数であって、表示する数字ではありません。配列の要素[箱番号]を指定する変数となります。

配列を使って、このように表示されれば正解です。

この段階では、表示結果は、Step2と全く同じになります。

Step4：配列の内容を乱数で入れ替える

- Step3で、作成した配列の内容を、乱数を用いて入れ替える。
- 乱数で配列の中身の入れ替えは、shuffle( )という関数を作成して行う。



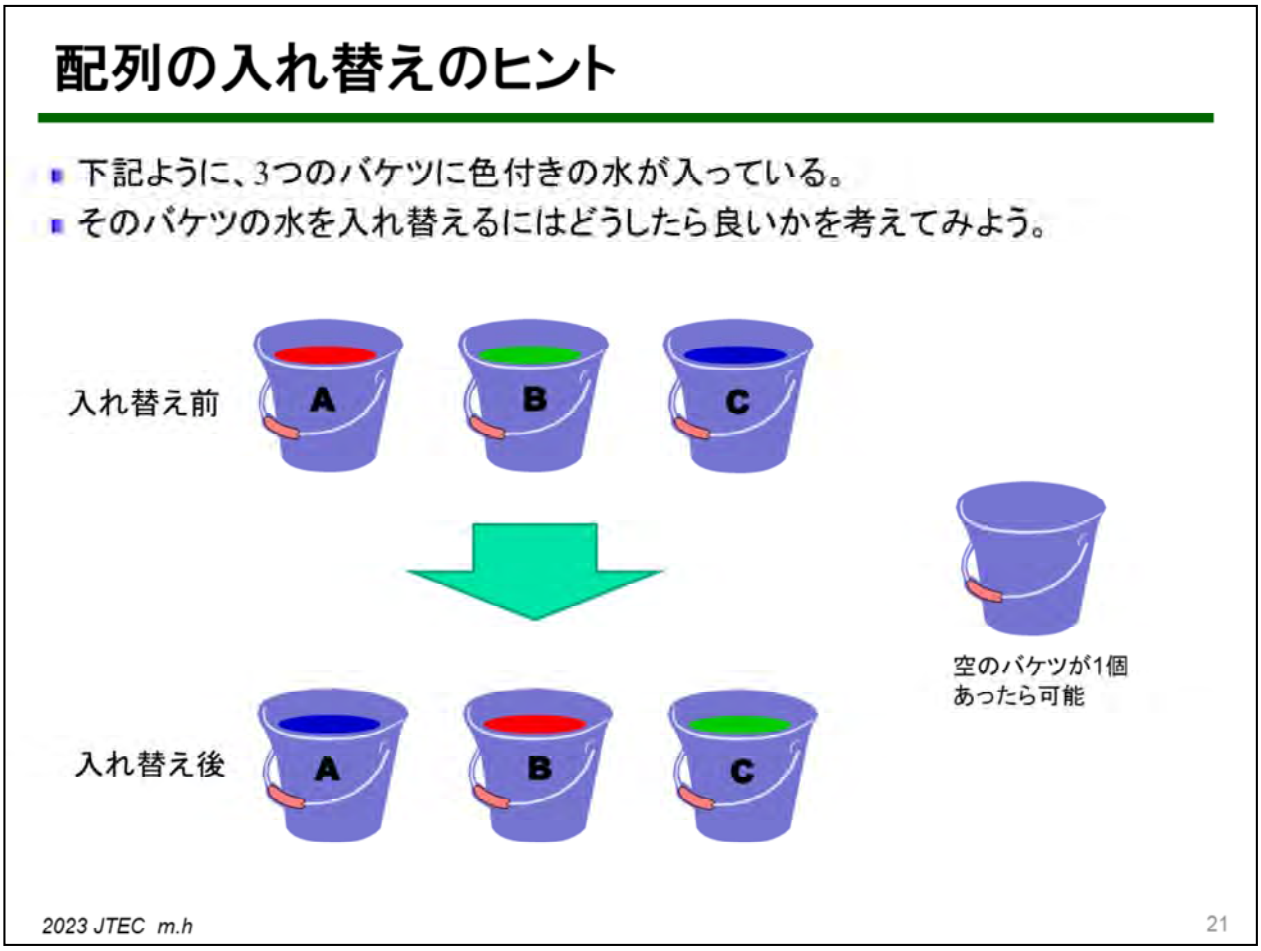
・乱数は、rand関数で得られる。 rand関数を使うには、#include <stdlib.h>が必要。
・単に、1～9まで乱数を求めると、同じ数字が重複する場合がある。
・そこで、予め1～9が入っている配列の内容を入れ替えることで、重複しない1～9の数字を求めることができる。
・また、rand関数は、何もしないで使用すると乱数のシード(種)が同じため、毎回同じパターンの乱数になる。
・そこで、最初に、srand((unsigned) time(NULL)); を実行して、乱数計算のシード(種)を変更する。
・これは、現在の時間を種として設定するものである。time関数を使うためには、#include <time.h>が必要。
・そして、rand()で求めた乱数から、9の余りを求めると、0～8の数字が得られる。
・その0～8の数字を使って、配列の内容を入れ替える。

2023 JTEC m.h 20

Step4は、Step3でやった配列の表示で、配列の中を乱数を使って入れ替えてみます。
Step3ができていれば、九九の表を表示するプログラムの部分に一切変更をしません。
九九の表を表示する前に、配列の中を入れ替える関数を呼び出すだけです。
仮に、配列の中を入れ替える関数を、shuffle( )とすると、
配列は、縦、横の2つあるので、
shuffle(row_tbl); ← 縦の配列を入れ替える
shuffle(col_tbl); ← 横の配列を入れ替える
の2行を追加します。

乱数は、rand( )関数で求めます。rand関数を使うには、#include <stdlib.h>が必要ですので、プログラムの最初に追加します。rand( )は、int型の整数値を返してきます。このままでは、大きな数字も返ってきます。今回、欲しい数字は、0～8までです。ということは、MAXSIZE で余りを求めれば、0～8までの数値が求まります。
乱数と言っても、中でサイコロを振っているわけではありません。rand( )は、種となるある数字から計算で求めています。その種を設定するのが、srand( )関数です。srandの「s」は、seed(種)の意味です。例えば、srand(1000)としたら、最初の種が1000となり、そこから計算で乱数を求めていきます。
種を設定することで、毎回同じ順で乱数が生成されるので、デバッグの時などは便利です。

ただ、今回は、不作為の乱数にしたいので、種の値にtime() 関数を使って現在の時間を与えます。time()関数は、ある基準時間(1970年1月1日0時0分0秒)からの経過時間になっているため、二度と同じ時間は現れません。そのため、不作為の乱数になります。実際は、
srand((unsigned) time(NULL)); とします。time() 関数を使うには、#include <time.h>が必要ですので、
stdlib.h 同様に最初に追加します。
また、srand() は、最初に1回だけ実行します。よく、shuffle( )の中でsrand( )を実行する人がいますが、そうすると、縦と横が同じ値になってしまいます。何故かというと、コンピュータはとても速いので、縦と横を2回呼んでも、同じ時間を取得してしまうためです。



この図は、配列の入れ替えのヒントを表したものです。

まず、3個のバケツに、それぞれ色付きの水か入っているとします。そして、このバケツの中の水を入れ替えることを考えて見てください。

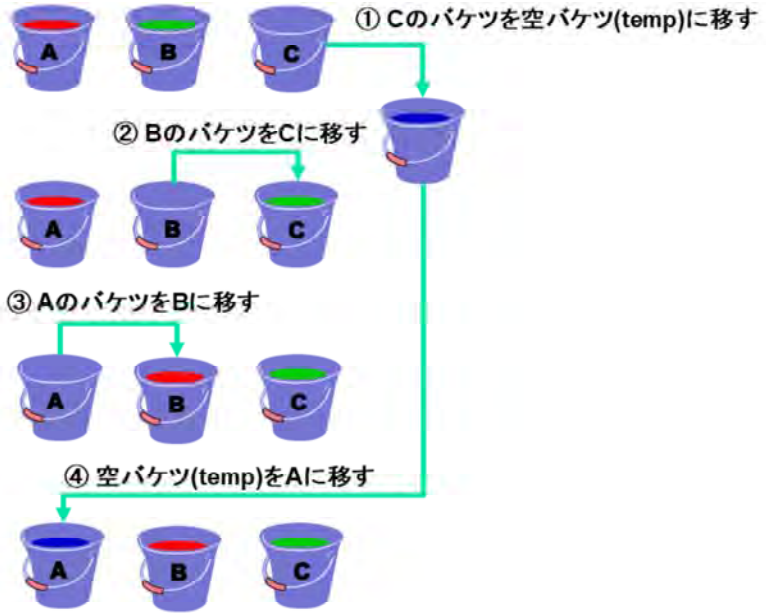
バケツが3個だけしかないなら、入れ替えは不可能です。、でも、そこに空バケツを1つ用意すれば、簡単に入れ替えることができることがわかると思います。

同様なことをプログラムで実現すれば良いことになります。

空バケツとは。一時的に内容を保存する変数の意味します。

配列の入れ替えのヒント

- 空バケツをtempという名前にしたとすると、



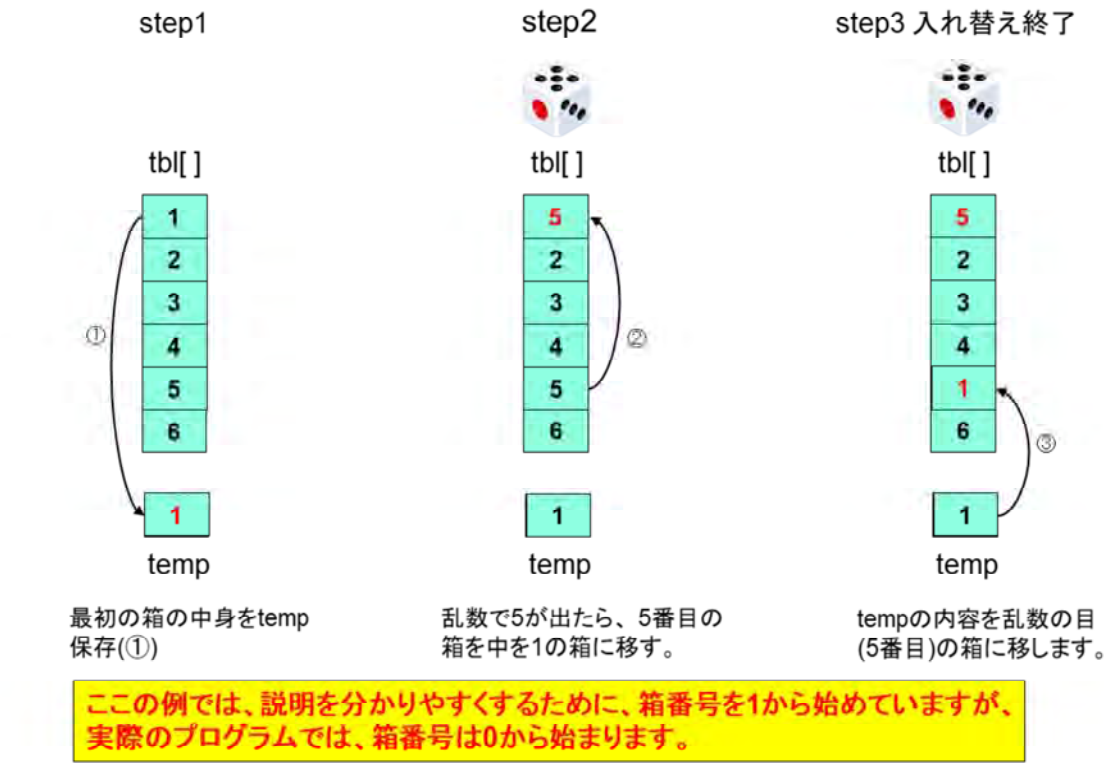
2023 JTEC m.h 22

この空バケツを、仮にtempという名前にしたとすると、
① Cのバケツの水を空バケツ(temp)に移します。
② Bのバケツの水を空になったCのバケツに移します。
③ Aのバケツの水を空になったBのバケツに移します。
④ 最後に、最初に用意した空バケツ(temp)を空になったAのバケツに移します。

以上の手順で、3個のバケツの水の入れ替えができます。

プログラムでも同様のことをします。
ただ、この例では、いきなり3個のバケツをすべて入れ替えていますが、実際のプログラムでは、③の処理で、空バケツ(temp)の中を空になったBのバケツに移すという3ステップを配列の個数だけ繰り返します。

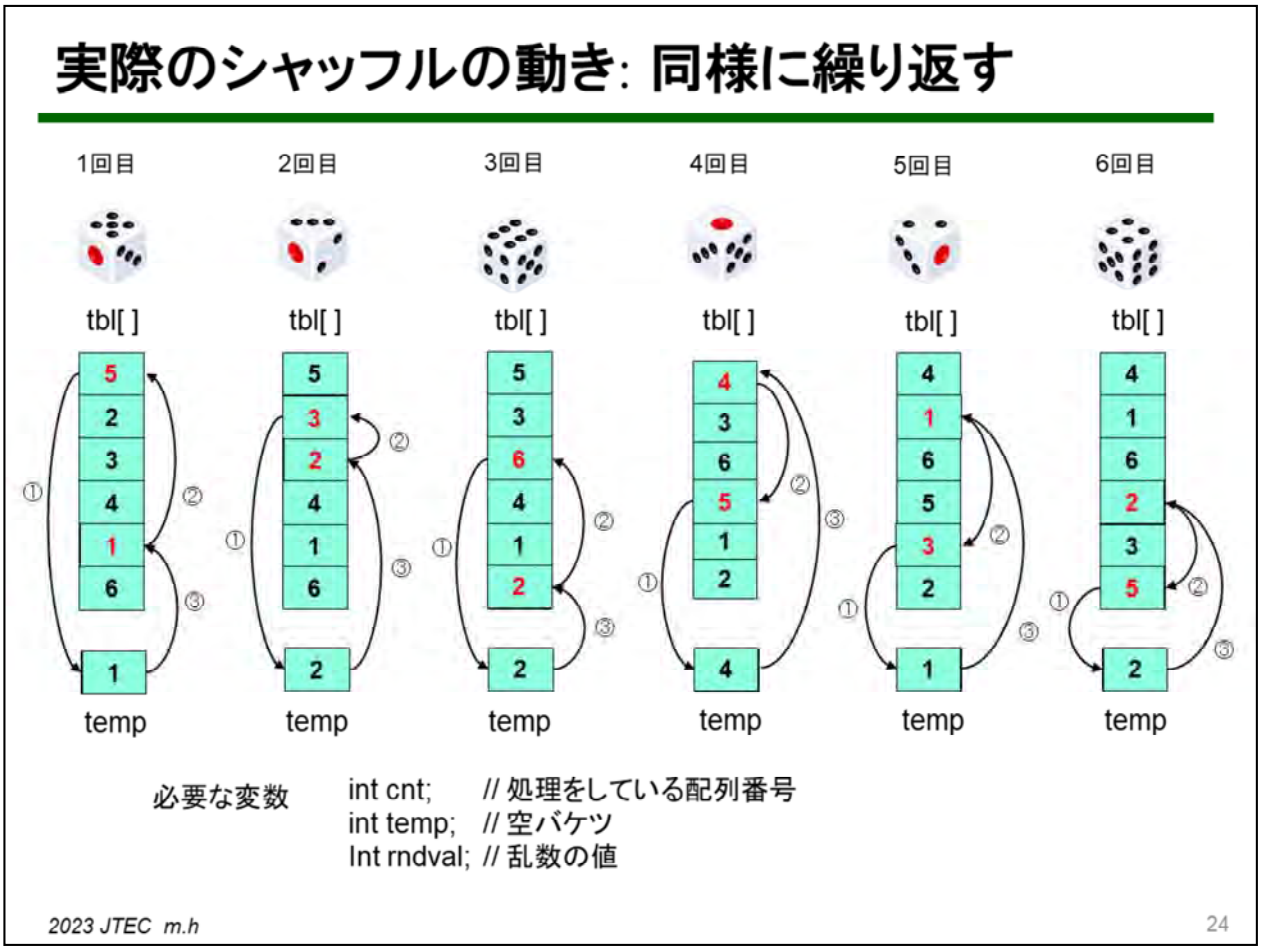
配列の入れ替えの動作



これは、配列の中の1個の要素の入れ替えを示したものです。この図で示すように、3ステップで1個の要素の入れ替えが完了します。

この操作を、配列の箱番号0～配列の個数だけ実施すれば、すべての配列の中が入れ替われます。

これは、最初か重複しない数字が入っている配列の中の入れ替えですので数字が、重複することはありません。また、仮に同じ乱数がでても直接は関係しません。



これは、サイコロの目に応じて、6個の配列の中を入れ替えを示したものです。

実際は、下記のようになります。

For (cnt = 0 ; cnt < 配列の個数 ; cnt++){　← 配列の個数分だけループする

　　①の処理

　　乱数の生成

　　②の処理

　　③の処理

}

新たに関数を作成する

2023 JTEC m.h 25

突然、関数の作成と言っても、どうしら良いかわからない人もいるかと思います。

新たに関数を作成するヒントとして、「C言語の基礎の基礎」のテキストから一部抜粋して説明をしています。

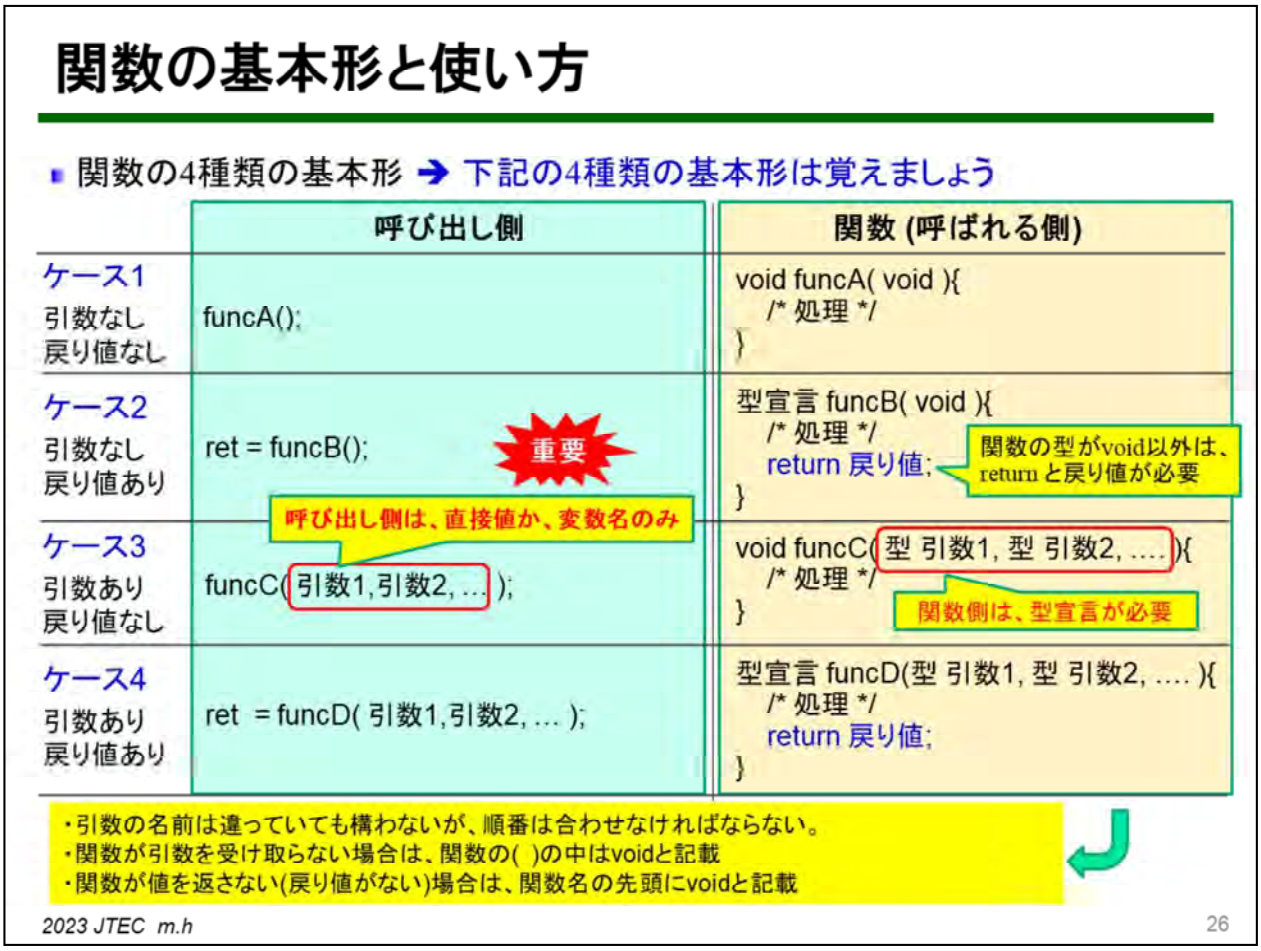
ここでの説明が不足であれば、「C言語の基礎の基礎」のテキストを見てください。

プログラムは、1つの大きなプログラムで作るのではなく、機能や役割ごとに処理を分割し、それらを組み合わせて作るのが基本です。

C言語では、機能や役割に分割して処理をさせるのに関数を作成します。

ここでは、その関数の作成方法について説明します。

C言語は、複数の関数を組み合わせて、結果的には大きなプログラムにします。



この表は、4種類の関数の基本形について、呼び出し側と関数(呼ばれる側)とを整理したものです。

呼び出し側は、関数名と引数がある場合は ( )の中に引数を書きます。

引数が複数ある場合は、「,」で区切って並べます。

このとき、引数が変数の場合でも、変数の型は書きません。変数名だけにします。

初心者が間違う例とし、呼び出し側の引数の変数に型まで書いてしまうことがあります。

一方、関数側(呼ばれる側)は、色々と必要です。

まず、関数に型が必要です。戻り値は、返す内容によって、int型だったり、char型だったり、あるいはポインタの場合もあります。変数の定義と同じように、関数名の前に型を書きます。

そして、値を返さない場合は、void と書きます。voidとは、空っぽという意味があります。

また、引数には、変数定義の同じように、「型 変数名」を書きます。← ここが、重要です。

C言語の関数は、呼び出すときは変数名だけ、受け取る側(関数)で型宣言が必要です。

引数がない場合は、関数名の( )の中にvoidと書きます。

内部的な動作として、関数を呼び出すとき、引数は「スタックに積まれたアドレス」が渡されます。そのため、受け取った側(関数)で、そのアドレスをどのように解釈するかということになります。

「スタックに積まれたアドレス」とは、ちょっと難しいです。本テキストの最後にある「再帰呼び出し」を参考にしてください。

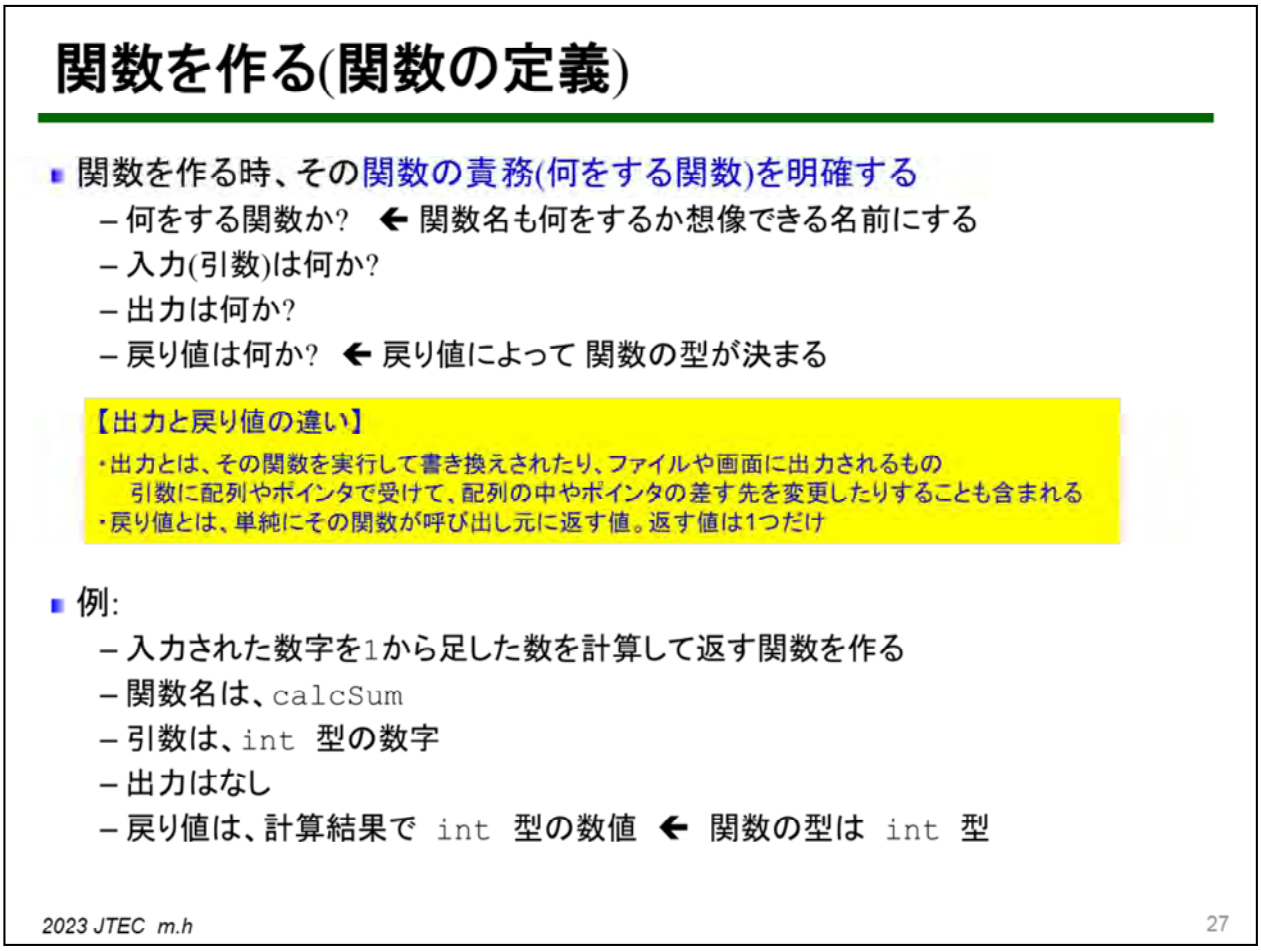
例えば、char ary[5][2] の2次元配列を引数にする場合、

関数を呼び出すときは、単にfunc(ary);と書きます。

一方、関数側では、void func(char ary[ ][ ]) として2次元配列で受けることになります。

このとき、関数側は、2次元配列を、5 x 2として受け取るべきか、2 x 5 で受け取るべきかが分かりません。

、そのため、受け取る側で、 void func(char ary[5][2]); のように変数の定義が必要になるのです。



それでは、実際に関数をどのように作成するかの説明をします。関数を作成するとき、重要なのが、その関数の責務(何をする関数)を明確にすることです。

(1) 何をする関数か? ← 関数名も何をするか想像できる名前にする

(2) 入力(引数)は何か?

(3) 出力は何か? ← 関数を実行したことの結果

(4) 戻り値は何か? ← 戻り値によって 関数の型が決まる

関数名も、可読性の観点から、変数名同様に、何をする関数が想像できる名前にします。

【出力と戻り値の違い】

・出力とは、その関数を実行して書き換えされたり、ファイルや画面に出力されるもので、その関数を実行した結果になります(引数に配列やポインタで受けて、配列の中やポインタが指し示す先を変更したりすることも含まれます)。

・戻り値とは、単純にその関数が呼び出し元に返す値です。返す値は1つだけで、複数の値を返すことはできません。

実際の例として、入力された数字を1から足した数を計算して返す関数を作る場合、

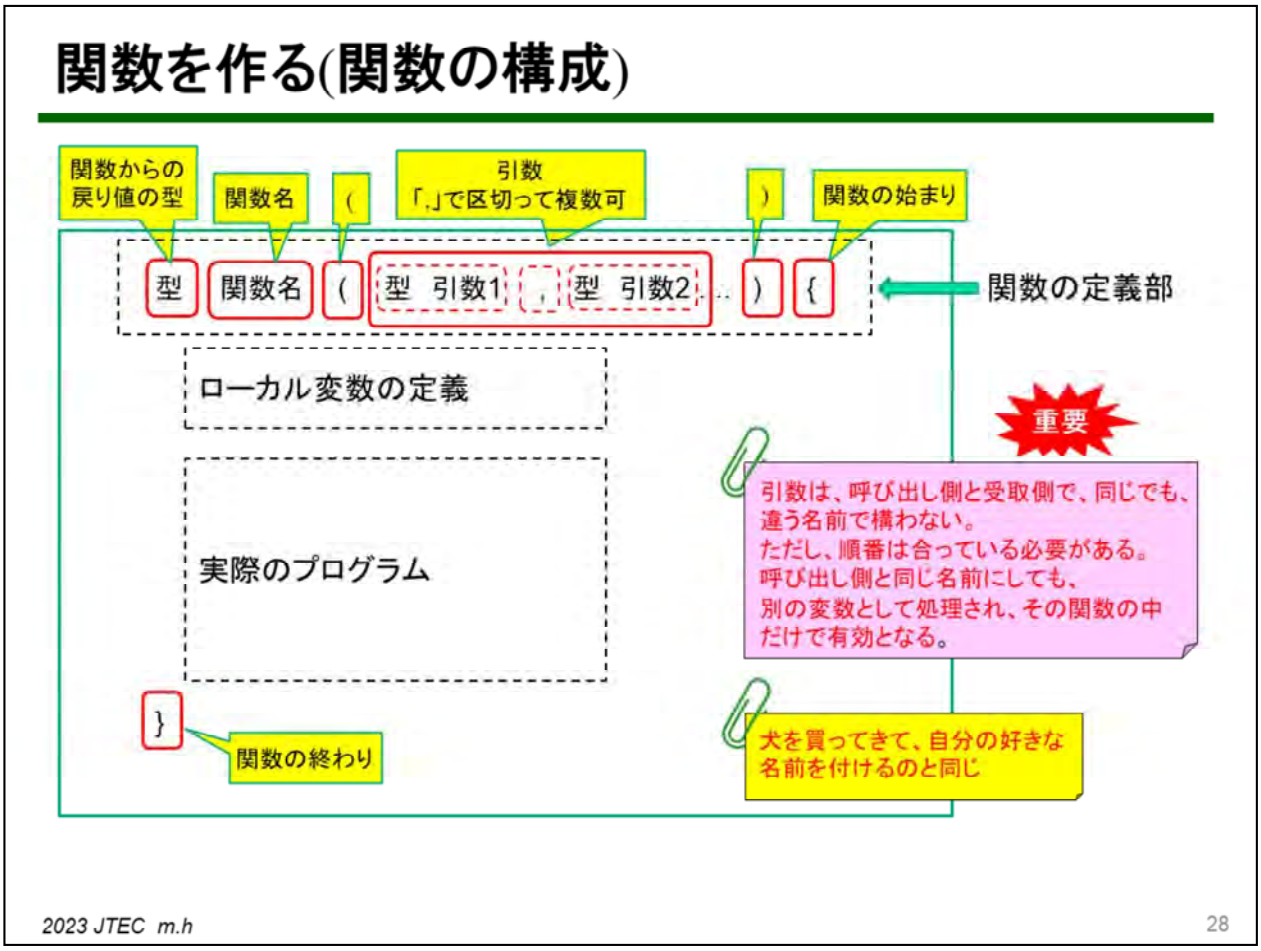
責務: 入力された数字を1から足した数を計算して返す

関数名:calcSum

引数: int 型の数字

出力: なし

戻り値: 計算結果で int 型の数値 ← 関数の型は int 型



責務:　入力された数字を1から足した数を計算して返す

関数名:calcSum

引数:　int 型の数字

出力:　なし

戻り値: 計算結果で int　型の数値　← 関数の型は int 型

にしたがって、関数の定義をすると下記のようになります。

```
Int  calcSum ( int  num ){
      ローカル変数の定義
          :
      実際のプログラム
          :
          :
      return  値  // 戻り値
}
```

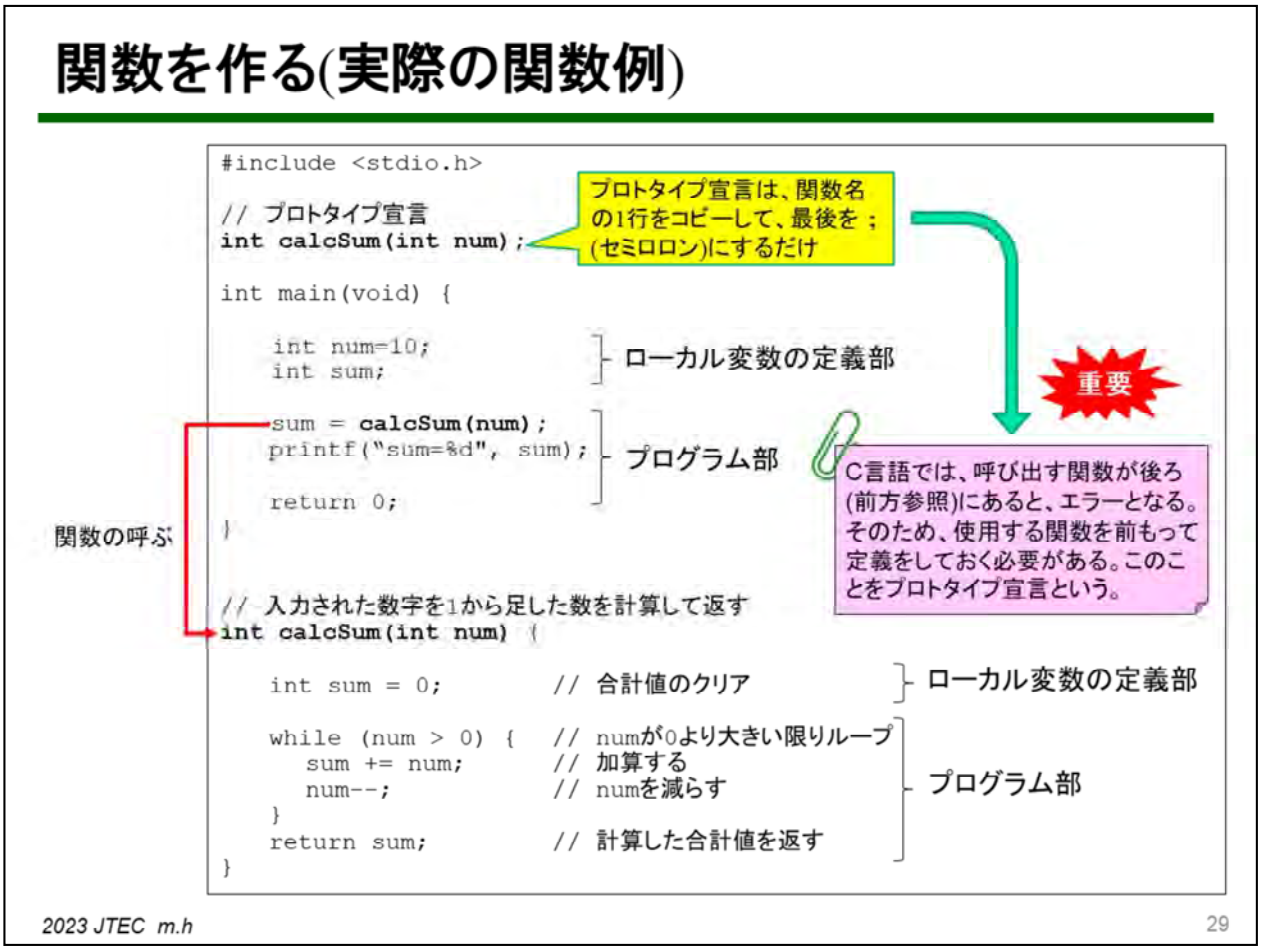
引数は、呼び出し側と関数側で、同じ名前でも別の名前でも構いません。

同じ名前だとしても違う変数と処理されます。これは、変数の有効範囲の話して、とても重要なことなのであとで詳しく説明します。

ペットショップから犬を買ってきて、自分の好きな名前を付けるのと同じことです。

また、関数とは、ホテルの部屋と考えると分かりやすいです。

ホテルの部屋の中で定義した変数は、その部屋の中だけて有効で、部屋を出ると自動的に消滅してしまいます(ローカル変数の場合)。



実際のコーディング例です。ここの例の中で、

// プロトタイプ宣言
int calcSum(int num);
とあります。

C言語では、関数を呼び出すとき、予め関数名か登録(定義)されていないと、不明な関数名としてエラーになってしまいます。そのたろ、その関数名を予め定義することをプロトタイプ宣言と言います。

よく、ネット上のサンプルプログラムで、main() が一番最後になっている例を見かけますが、これは、このプロトタイプ宣言を省略するためです。
プログラムの可読性の観点から見ると、main()関数を一番最初に持ってき、そのあと使用する関数を並べた方が分かりやすいです。

プロトタイプ宣言は簡単で、実際の関数名の1行をコピーして、関数名の最後にある { を削除して、: にするだけです。上記の例では、
`int calcSum(int num) {` ➔ `int calcSum(int num);`

shuffle() 関数を作る

- shuffle( )関数:
  - 機能: 引数で与えられた1次元配列の中を乱数で入れ替える。
  - 関数名: shuffle
  - 引数: char型の1次元配列 ← char tbl[ ]
  - 出力: 入れ替わった配列
  - 戻り値: なし ← 戻り値がないので、関数の型はvoidとなる
- 上記を元にすると、関数定義は下記のようになる

```
void shuffle(char tbl[ ]){      ← 1次元配列を引数として受け取る

}
```

- ローカル変数として、3個必要

```
int cnt;        // 処理をしている配列番号
int temp;       // 空バケツ
Int rndval;     // 乱数の値
```

2023 JTEC m.h 30

まず、shuffle( )関数の機能を明確にします。

機能: 引数で与えられた1次元配列の中を乱数で入れ替える。
関数名: shuffle
引数: char型の1次元配列 ← char tbl[ ]
出力: 入れ替わった配列
戻り値: なし ← 戻り値がないので、関数の型はvoidとなる

機能を元に、関数を定義すると下記のようになります。

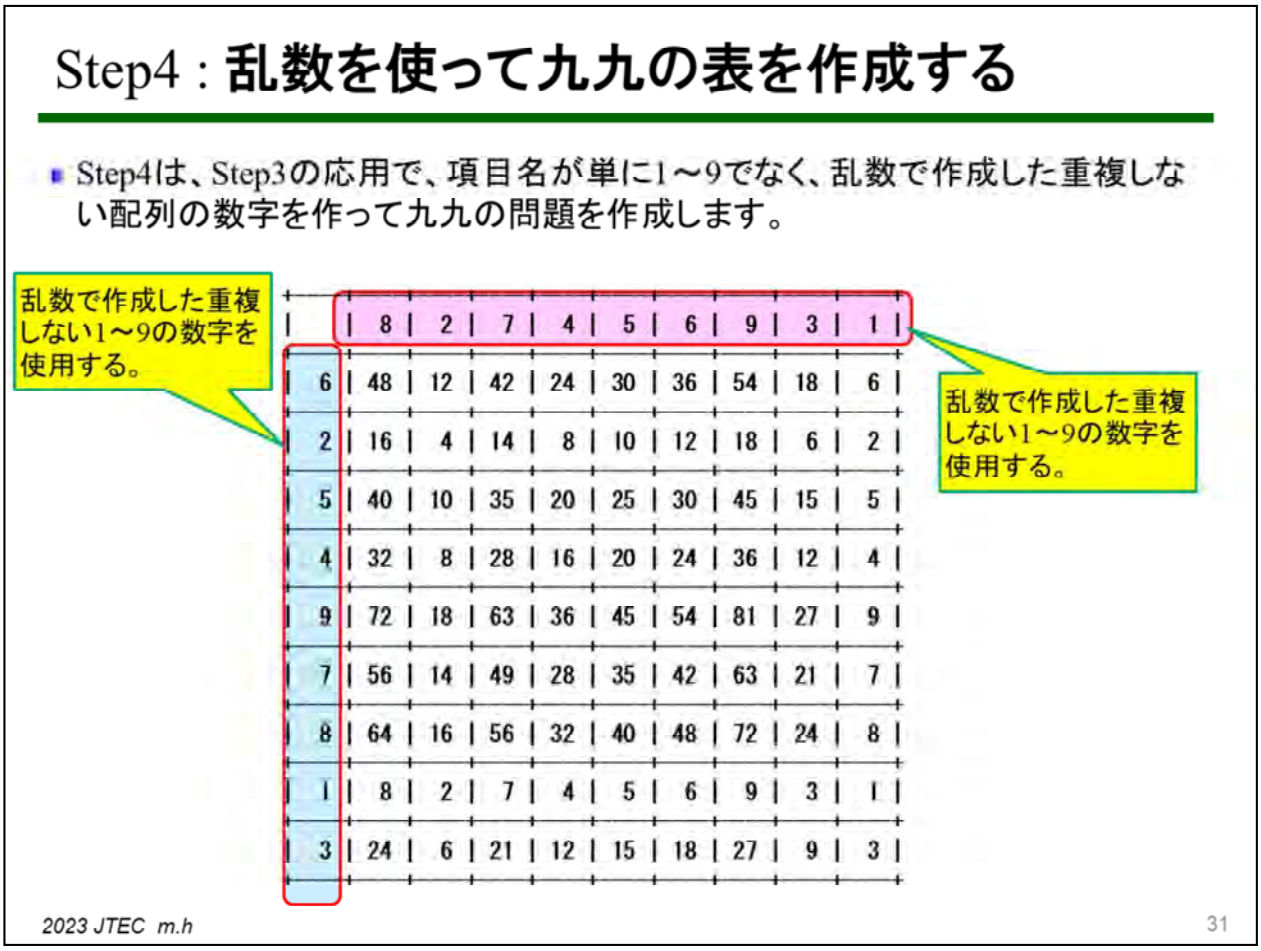
```
void shuffle(char tbl[ ]) {    ← 1次元配列を引数として受け取る
}
```

次のにローカル変数として、次の3個が必要になります。

```
int cnt;      // 処理をしている配列番号
int temp;     // 空バケツ
Int rndval;   // 乱数の値
```

その結果、shuffle関数は、次のような構成になります。

```
for (cnt = 0; cnt < MAXSIZE; cnt++){
    ①tbl[cnt]をtempに移す
    ②rndval にrand() 関数で(0～8)の乱数を取得する
    ②乱数で示す配列の中(tbl[rndval]を、cntが示す配列に移す
    ③乱数で示す配列に保存したtempの内容を移す
}
```



shuffle() 関数ができれば、この関数を呼ぶだけで、指定した配列の内容が入れ替えられます。

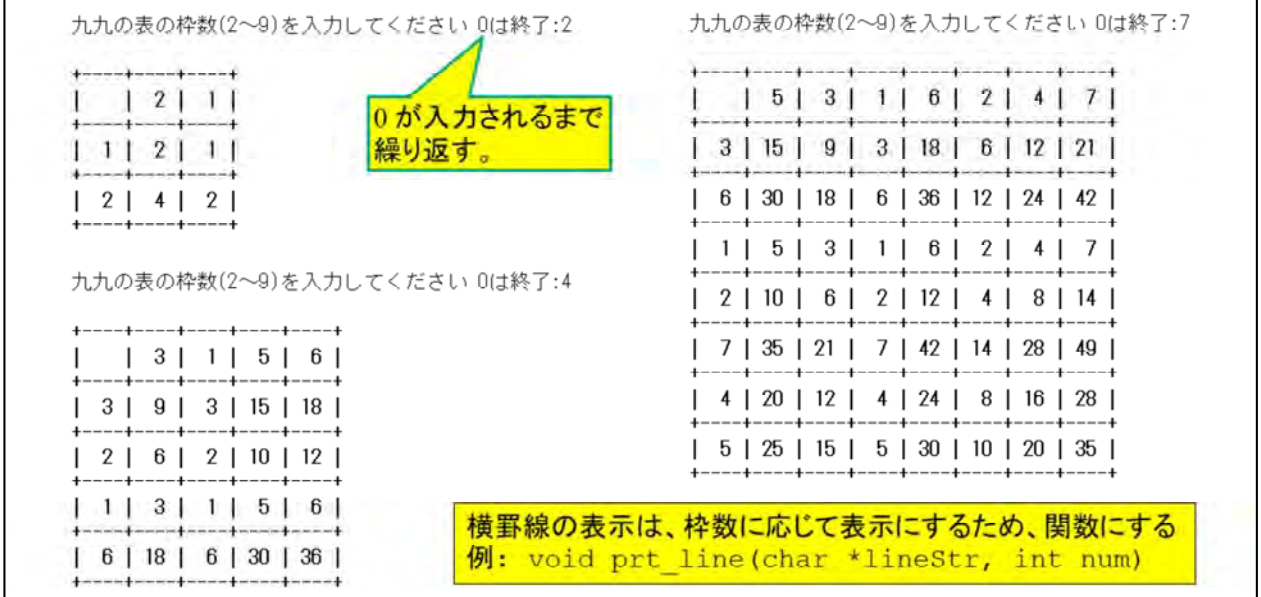
配列の名前が、仮に、row_tbl[]、col_tbl[]だとすると、
mainからは、Step3で作成した九九の表示の処理をする前に、下記の2行を追加するだけです。
shuffle(row_tbl);　← 縦の列の入れ替え
shuffle(col_tbl);　← 横の配列の入れ替え

ここから、Step3の処理になります。Step3の処理に変更ありません。

実行した結果、縦と横の項目の数字がバラバラになっていることを確認してください。

Step5：表示する枠数を入力して作成

- Step5はStep4の応用です。scanf()を使って数値を入力し、その数値の大きさの枠数で表示する。0が入力されるまで繰り返す。配列の大きさは縦(9)と横(9)のままで、画面表示だけを入力された数字の大きさの枠で表示する。



2023 JTEC m.h 32

入力された枠数に応じて表示をするには、横罫線も枠数に合わせなければなりません。そのため、横罫線を表示するための関数を作成します。

Step4までは、

printf(“+----+----+----+----+----+----+----+----+----+”);

で、横罫線を表示していました。

このprintf( )の部分を関数にします。

例えば、“+----” の文字列を定義しておいて、これを枠数に応じて繰り返し表示します。

そうすると、関数の仕様が見えてきます。作成した関数の再利用を考慮すると、引数は2つあったほうが良さそうです。つまり、「指定された文字列を、指定された数だけ表示する」ということになります。

関数の呼び出しは、

char *lineStr = “+----”;

pri_line(hlineStr, 入力された枠数); ← printf(“+---…+----+”);を置き換える

とします。

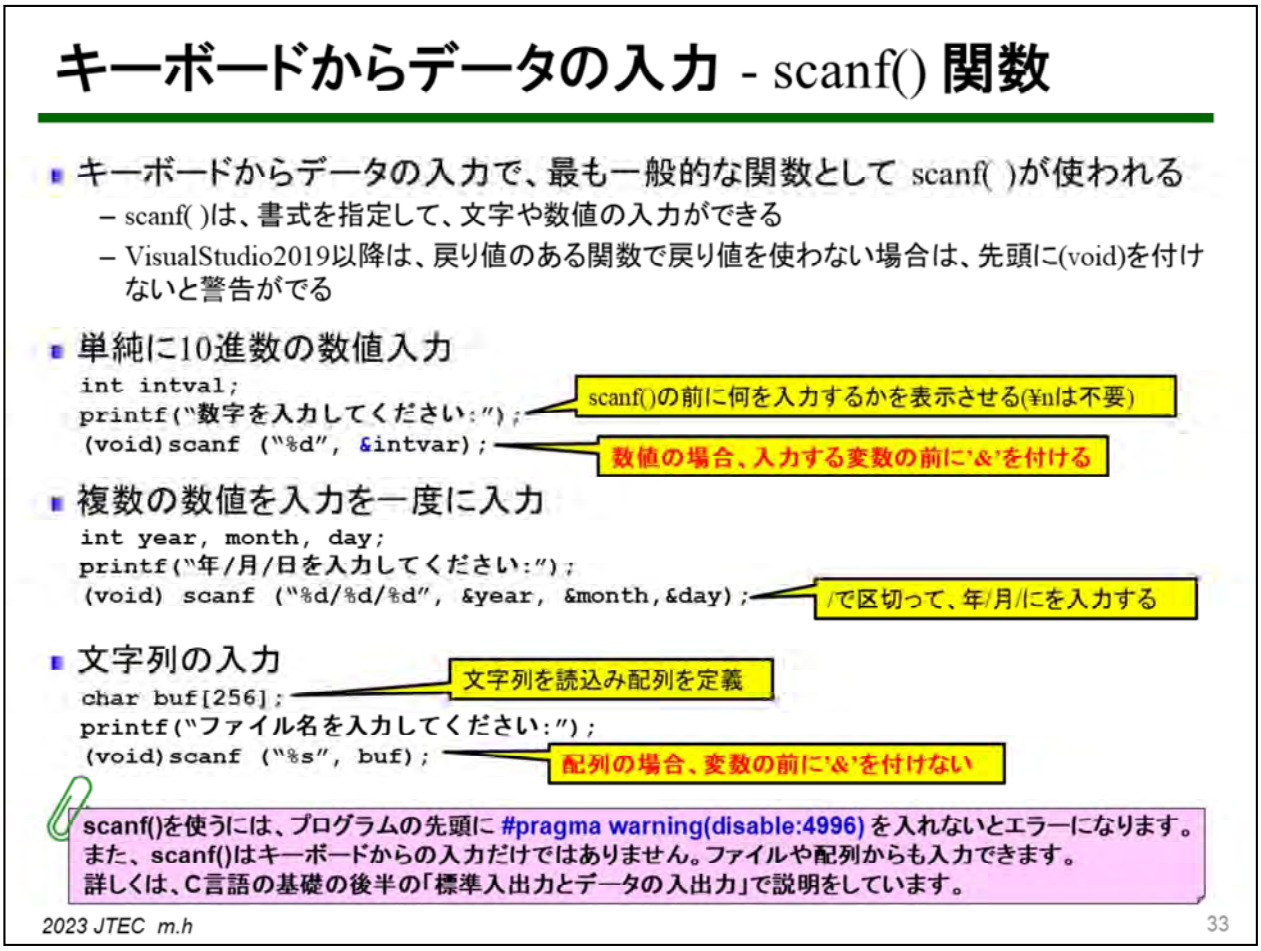
そして、実際の関数としては、

void prt_line(char *lineStr, int num){

　ここで、lineStrをnum回表示する ← for文でOK

　for文が終了したら、lineStrの最初の1文字を表示して改行する ← 1文字表示は、”%c”を使います。

}



キーボードからデータ入力するには、色々な関数や方法があります。最も一般的な関数として、scanf( )が使われます。scanf( )は、書式を指定して、文字や数値の入力ができます。

単純に10進数の数値入力する場合、下記のようにします。

```
int intval;                          ← 変数の定義
printf(“数字を入力してください:”);  ← 同じ行にカーソルを表示するため¥n付けないほうが良い
scanf (“%d”, &intvar);               ← 入力する変数の前に、「&」記号を付けます。
```

複数の数値を入力を一度に入力したい場合は、下記のように書式指定に区切り記号を入れます。

```
int year, month, day;
printf(“年/月/日を入力してください:”);
scanf (“%d/%d/%d”, &year,&month,&day);
```

この例は、’/’で区切っているので、実際のキーボートから入力は、「2021/4/1」と入力すると、各変数に年月日が入ります。

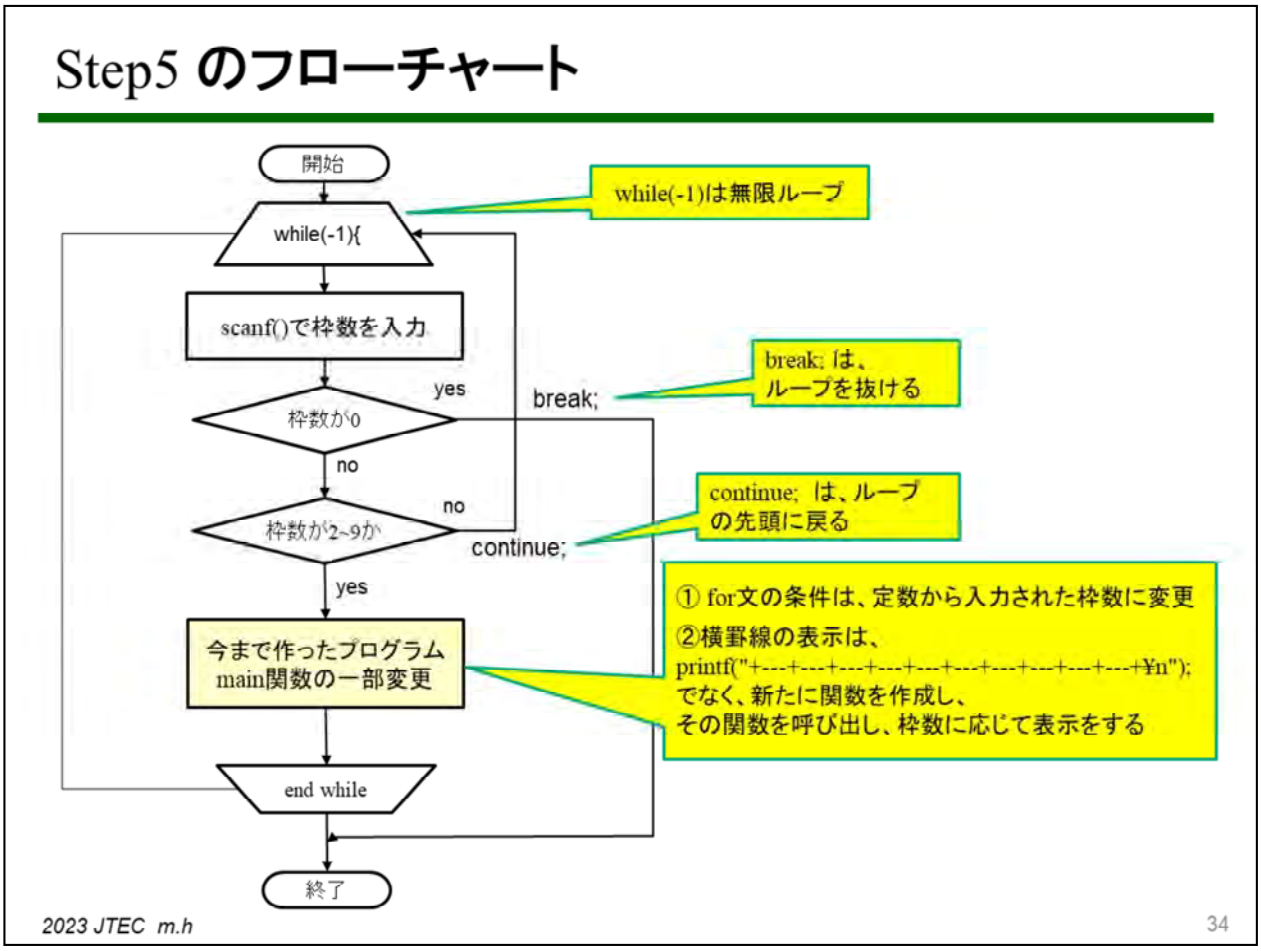
‘/’を’.’にして、“%d.%d.%d”とすると、「2021.4.1」と入力します。区切り記号はスペースでも構いません。その場合は、“%d %d %d”として、「2021 4 1」と入力します。

文字列を入力する場合は、下記のように書式指定を”%s”にします。

```
char buf[256];
printf(“ファイル名を入力してください:”);
scanf (“%s”, buf);
```

文字列を入力するために、文字列が十分入るchar型の1次元配列を定義します。

そして、ここで注意をするとは、配列の変数名の先頭には、「&」記号は付けません。これは、もともと配列の引数は、アドレスが渡されるためです。



これは、Step5のフローチャートです。クリーム色の四角は、今までやってきたプログラムそのままです。
変更点として、九九の表示をするfor文の条件が、MAXSIZEの定数から、scanf( )で入力して枠数になります。 shuffleは、枠数に関係なく、すべての配列の入れ替えをします。

Step5が完成したら、最初に定義した、
#define MAXSIZE 9 を、例えば、#define MAXSIZE 12 に変更して正しく動作するようにします。
このとき、2カ所変更が必要です。

(1) row_tbl[ ]とcol_tbl[ ]の初期化

今までは、2つの配列の初期化を変数の定義時に行っていました。

```
char row_tbl[9] = { 1,2,3,4,5,6,7,8,9 };
char col_tbl[9] = { 1,2,3,4,5,6,7,8,9 };
```

これをやめて、for文を使ってプログラムで、row_tbl[ ]とcol_tbl[ ]に、1～MAXSIZEの数字を入れて初期化します。

(2) 最大入力可能な枠数を表示する

”九九の表の枠数(2～9)を入力してください 0は終了:”の最大値「9」は、
MAXSIZEの値を表示するようにします。
どうすれば良いか分かりますか? ➔ 「9」を{%d」に変えて、MAXSIZEの値を表示するようにします。

```
printf(”九九の表の枠数(2～%d)を入力してください 0は終了:”, MAXSIZE);
```

とすれば、良いですね。
この例では、最大枠数、「12」を入力して、12x12枠で表示できれば正しくコーディングされています。