

プログラミング実習

第2回：VSCodeの設定，分岐，繰り返し

清水 哲也（ shimizu@info.shonan-it.ac.jp ）

今回の授業内容

- 授業内容について説明
- VSCodeの設定
- 復習
- 授業課題
- 宿題

授業内容について説明

授業内容について説明

清水クラスの特徴

プログラミング実習のクラスの中で（多分）1番難しい内容を扱います。
できる限りModづくりに時間を使いたいので復習部分は解説しません。

プログラミング基礎の復習について

今回は教科書の第1章～第4章までの内容を扱います。
具体的には、変数、読み込みと表示、演算、型、if文、switch文、do while文、for文、多重ループです。

VSCodeの設定

VSCodeの設定

Min WのPATH確認 part.1

MinGWのPATH設定ができているかを確認します.

「コマンドプロンプト」か「ターミナル」を起動してください.
次のコマンドを入力してEnterキーを押してください.

```
gcc -v
```

VSCodeの設定

Min WのPATH確認 part.2

実行結果が以下のようになっていればPATH設定が正しくされています。

```
C:\Users\Shimizu>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=c:/mingw/bin/./libexec/gcc/mingw32/9.2.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-9.2.0/configure --build=x86_64-pc-linux-gnu --host=mingw32 --target=mingw32 --disable-win32-registry --with-arch=i586 --with-tune=generic --enable-static --enable-shared --enable-threads --enable-languages=c,c++,objc,obj-c++,fortran,ada --with-dwarf2 --disable-sjlj-exceptions --enable-version-specific-runtime-libs --enable-libgomp --disable-libvtv --with-libiconv-prefix=/mingw --with-libintl-prefix=/mingw --enable-libstdcxx-debug --disable-build-format-warnings --prefix=/mingw --with-gmp=/mingw --with-mpfr=/mingw --with-mpc=/mingw --with-isl=/mingw --enable-nls --with-pkgversion='MinGW.org GCC Build-2'
gcc version 9.2.0 (MinGW.org GCC Build-2)
```

最後の行に `gcc version 9.2.0 (MinGW.org GCC build-2)` などと書かれていると思います。

(数字は `9.2.0` でなくでも大丈夫です。

設定の参考サイト：<https://www.javadrive.jp/cstart/install/index6.html#section3>

VSCodeの設定

VSCodeのターミナルから標準入力を使えるようにする

- 「拡張機能」から「Code Runner」を探します
- 「Code Runner」の右側にある歯車のマークを押します
- 「拡張機能の設定」をクリックします
- 設定画面が表示されるので「Run in Terminal」にチェックをいれます

これで、`scanf();`などの標準入力がターミナルから行うことができます

VSCodeの設定

設定が完了したので、動作確認をします。

ファイルを新規作成してファイル名を **HelloWorld.c** とします。

```
#include<stdio.h>

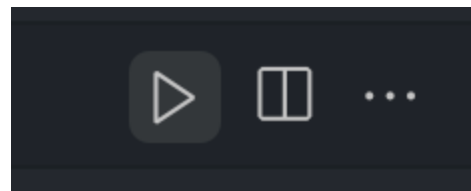
int main(void)
{
    printf("Hello World!\n");

    return 0;
}
```

これを書いてください。

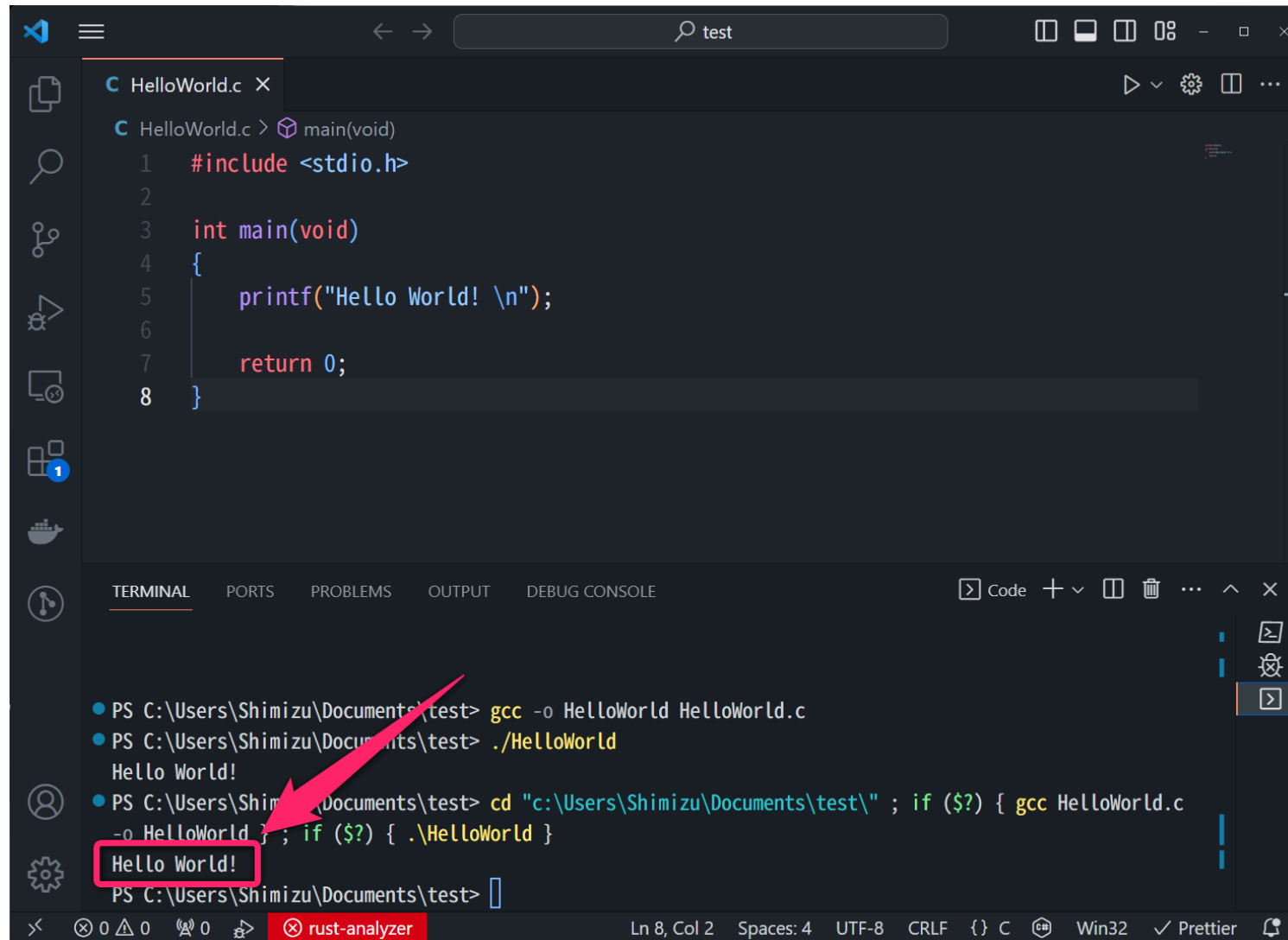
VSCodeの設定

VSCodeの右上にある三角マークを押して実行してみましょう



VSCodeの設定

VSCodeの下にターミナルが起動して実行結果が表示されると思います



The screenshot shows the Visual Studio Code interface. The editor window displays a C program named `HelloWorld.c` with the following code:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello World! \n");
6
7     return 0;
8 }
```

The bottom panel shows the **TERMINAL** tab with the following commands and output:

```
PS C:\Users\Shimizu\Documents\test> gcc -o HelloWorld HelloWorld.c
PS C:\Users\Shimizu\Documents\test> ./HelloWorld
Hello World!
PS C:\Users\Shimizu\Documents\test> cd "c:\Users\Shimizu\Documents\test\" ; if ($?) { gcc HelloWorld.c -o HelloWorld } ; if ($?) { ./HelloWorld }
Hello World!
PS C:\Users\Shimizu\Documents\test>
```

A red arrow points to the `Hello World!` output line, which is also highlighted with a red box. The status bar at the bottom indicates the file is `Ln 8, Col 2`, using `UTF-8` encoding, and has `Spaces: 4`.

復習

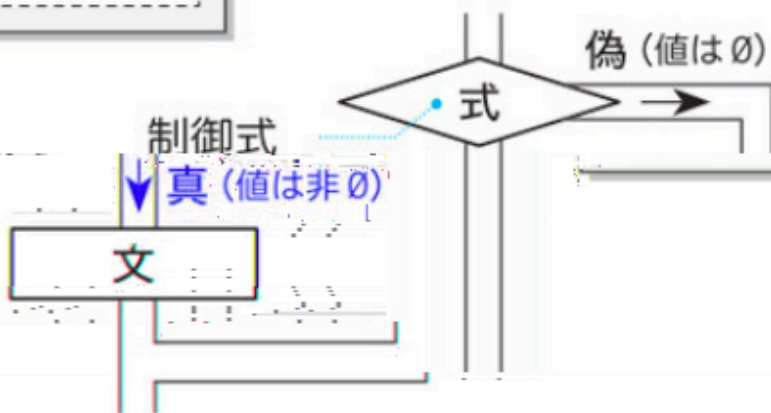
復習：if 文

ある条件が成立したときにのみ処理を行うことができる分岐です。

● if 文

if (式)
文

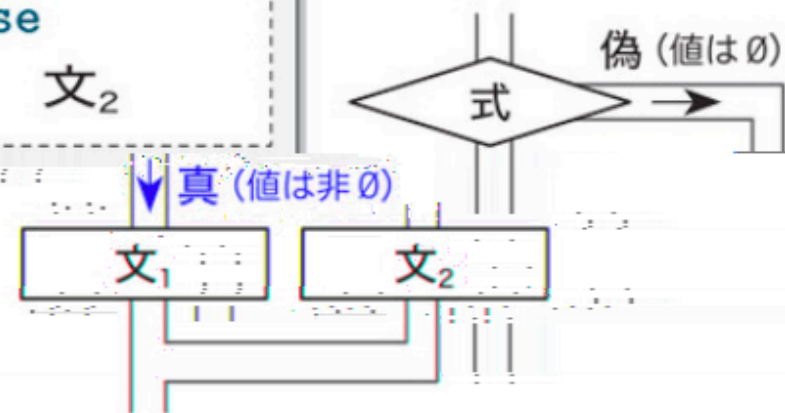
式を評価した値が非0であれば文を実行する



● if 文 (else 付き)

if (式)
文₁
else
文₂

式を評価した値が非0であれば文₁を実行し、0であれば文₂を実行する



復習：条件式評価

条件式を評価する.

- 等価演算子
- 関係演算子
- 論理演算子

などなど

復習：条件式の評価

等価演算子

演算子	例	意味
<code>==</code>	<code>a == b</code>	<code>a</code> と <code>b</code> の値が等しければ <code>1</code> , そうでなければ <code>0</code>
<code>!=</code>	<code>a != b</code>	<code>a</code> と <code>b</code> の値が等しくなければ <code>1</code> , そうでなければ <code>0</code>

復習：条件式の評価

関係演算子

演算子	例	意味
<	<code>a < b</code>	<code>a</code> が <code>b</code> よりも小さければ <code>1</code> , そうでなければ <code>0</code>
>	<code>a > b</code>	<code>a</code> が <code>b</code> よりも大きければ <code>1</code> , そうでなければ <code>0</code>
<=	<code>a <= b</code>	<code>a</code> が <code>b</code> 以下であれば <code>1</code> , そうでなければ <code>0</code>
>=	<code>a >= b</code>	<code>a</code> が <code>b</code> 以上であれば <code>1</code> , そうでなければ <code>0</code>

復習：条件式の評価

論理演算子

演算子	例	意味
&&	a && b	a と b の値がいずれも非 0 であれば 1 , そうでなければ 0
	a b	a と b の値の一方でも非 0 であれば 1 , そうでなければ 0

復習：switch文

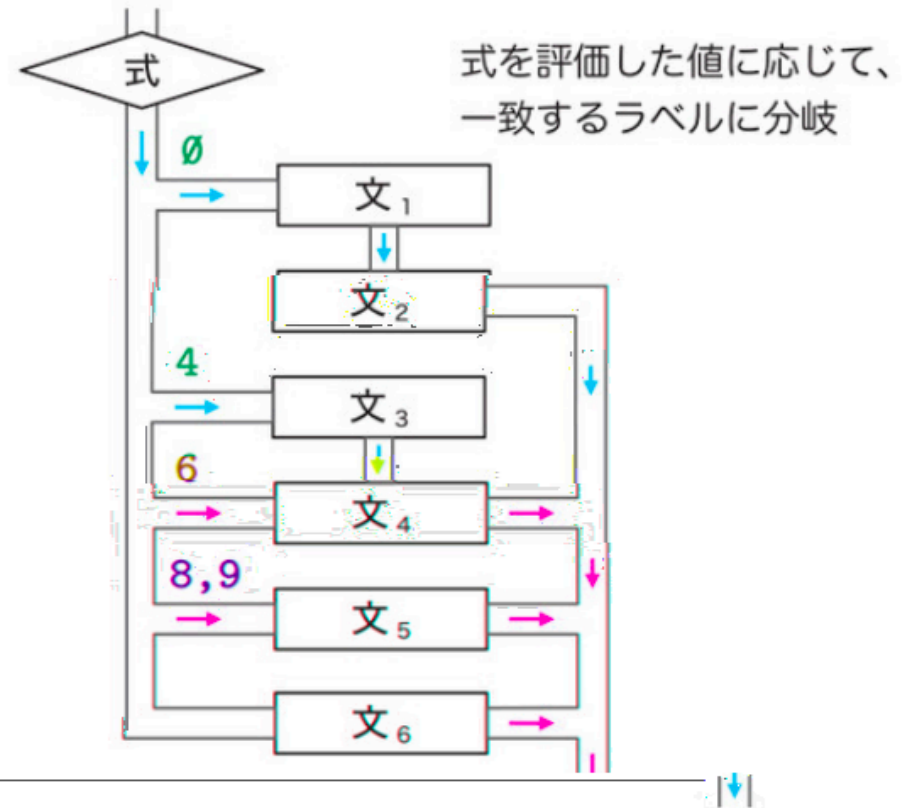
一つの式を評価した値に応じて、プログラムの流れを複数に分岐できます。

● switch 文

```
switch (条件) {  
  case 0 : 文1 文2 break;  
  case 4 : 文3  
  case 6 : 文4 break;  
  case 8 :  
  case 9 : 文5 break;  
  default: 文6 break;  
}
```

break 文

switch 文の実行を中断・終了する



復習：繰り返し文

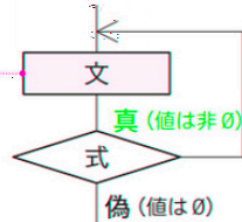
制御式を評価して条件に合えばループ本体が実行されます。
do文, while文, for文の総称です。

● do 文

```
do  
  文  
while (式);
```

式を評価した値が真である限り、文を繰り返し実行。

必ず一度は実行される

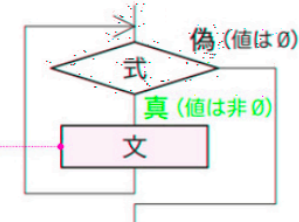


● while 文

```
while (式)  
  文
```

式を評価した値が真である限り、文を繰り返し実行。

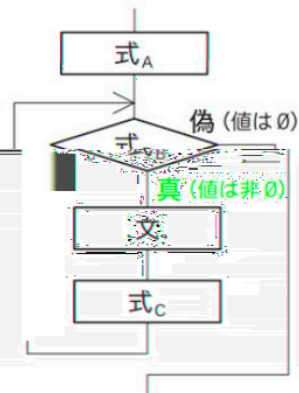
実行されるとは限らない



● for 文

```
for (式A; 式B; 式C)  
  文
```

式_Aを一度だけ評価・実行する（あるいは変数を宣言する）。
式_Bを評価した値が真である限り、
『文を実行して、式_Cを評価・実行する』
処理を繰り返す。



復習：do文, while文

do文：後判定繰り返しをします。ループ本体は少なくとも1回は必ず実行されます。

while文：前判定繰り返しをします。ループ本体は1回も実行されない可能性があります。

● do 文

```
do  
  文  
while (式);
```

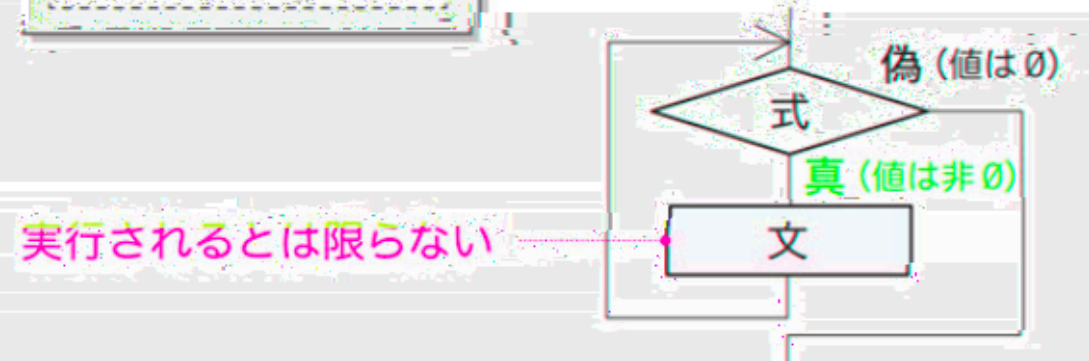
式を評価した値が真である限り、文を繰り返し実行。



● while 文

```
while (式)  
  文
```

式を評価した値が真である限り、文を繰り返し実行。



復習：for文

for文：前判定繰り返しをします。ループ本体は1回も実行されない可能性があります。単一のカウンタ用変数で制御する繰り返し方法です。

● for 文

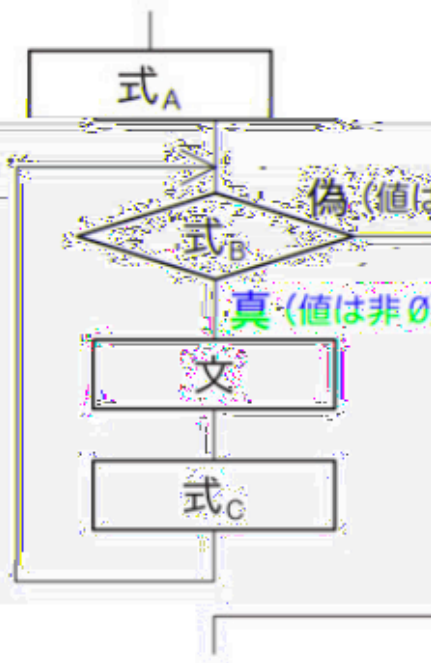
for (式_A; 式_B; 式_C)

文

式_Aを一度だけ評価・実行する（あるいは変数を宣言する）。

式_Bを評価した値が真である限り、

「文を実行して、式_Cを評価・実行する」処理を繰り返す。



授業課題

Moodleに授業課題ファイルがあります。
それを行ってください。

手順としては以下の通りです。

- プログラムを作成する
- 実行する
- プログラムをファイルに貼り付ける
- 実行結果のスクショをファイルに貼り付ける

課題は積極的に周りの人と相談したり，教えあったり，協力してください。

※答えをそのまま渡すのはやめましょう

宿題

Moodleに宿題ファイルがあります。
それをやってください。

手順と取り組みは授業課題と同じです。

提出期限は10月4日(水) 21:00まで