



# ce-1. C プログラミング入門

(C プログラミング応用) (全 1 4 回)

URL: <https://www.kkaneko.jp/pro/c/index.html>

金子邦彦



# C言語



- D. M. Ritchieが1970年にUNIXを記述するために作成した言語
  - メモリの操作など
- きめ細かな操作ができる
- ANSI により規格化、汎用性が高い
  - 広く利用されている
- 手続き型言語のひとつ
- コンパイラ型言語のひとつ

# メッセージ表示プログラム



Microsoft Visual Studio のデバッグ コンソール

```
Hello, World
```

```
C:\Users\user\source\repos\ConsoleApplication1\Debug\ConsoleAp  
ました。  
このウィンドウを閉じるには、任意のキーを押してください . . .
```

画面に表示されたメッセージ

```
#include "stdio.h"
```

```
int main()
```

```
{
```

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

```
}
```

# メッセージ表示プログラム



```
#include "stdio.h"
```

```
int main()  
{  
    printf("Hello, World¥n");  
}
```

1つの関数  
(関数名: main)

main 関数の中で、  
関数 printf を使っている

# メッセージ表示プログラム



```
#include "stdio.h"
```

関数名

```
int main()
```

```
{  
    printf("Hello, World¥n");  
}
```

**main** 関数  
の本体

個々の文の終了を示す  
(セミコロン)

# プログラムの流れ



プログラムの実行開始

↓  
メイン関数

```
#include "stdio.h"
int main()
{
    printf("Hello, World\n");
}
```

関数呼び出し

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

printf 関数

戻り

↓  
プログラムの実行終了



# 例題 1 . 棒グラフ

- 整数から, その長さだけの棒を表示する関数 bar を作る

例) 6 → \*\*\*\*\*

- 関数 bar を使って, 「整数を読み込んで, 読み込んだ長さの棒を表示するメイン関数を作る

# 棒グラフ



```
C:\> C:\Users\user\source\repos\ConsoleApp
```

```
len = 6  
*****
```

メッセージ「**len =**」が表示されるので、キーボードで「**6 Enter**」とすると、長さ6の棒グラフが表示される

実行結果の例



# プログラム実行順



- 普通, プログラム中の文は, 上から下へ順に実行される
- 関数呼び出しでは, 関数の先頭に「ジャンプ」する.

このことは, C言語が「手続き型言語」と言われる理由の1つ

関数呼び出しの例) **bar( len );**

- 呼び出された関数の中で return 文に出会うと, 関数呼び出しの場所に戻る.

# 棒グラフ



```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
```

bar関数

```
int main()
{
    int len;
    char buf[256];
    int ch;
    printf( "len = " );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%d¥n", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

main関数

複数の関数を含む  
プログラム

プログラム実行は  
main 関数（メイン  
関数）から始まる

# プログラム実行順



```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
```

戻り

メイン関数の先頭行  
がプログラム実行の始まり

```
int main()
{
    int len;
    char buf[256];
    int ch;
    ① printf( "len =" );
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%d¥n", &len );
    ⑦ bar( len );
    ⑧ ch = getchar();
    ⑨ ch = getchar();
    return 0;
}
```

関数呼び出し

メイン関数内の **return**  
がプログラム実行の終わり

# プログラムの流れ



プログラムの実行開始

↓  
メイン関数

関数呼び出し

**bar( len );**

↓  
プログラムの実行終了

```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}

int main()
{
    int len;
    int ch;
    printf( "len=" );
    scanf_s( "%d", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

↗  
bar 関数

↘  
**return;**

\* printf, scanf の  
呼び出しについては  
図では省略・12

# 変数



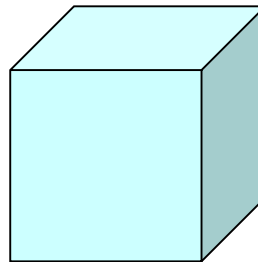
- 変数

データ（数値や文字）を入れるもの

- 変数名

英数字かアンダーバー（`_`）で作られる  
最初の文字には数字は使えない  
大文字と小文字を区別する

変数 `i`





```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
```

**len** をメモリエリア中に確保

変数 **i** をメモリエリア中に確保

```
int main()
{
    int len;
    char buf[256];
    int ch;
    printf( "len =" );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%d¥n", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

変数 **len, buf, ch**  
をメモリエリア中に確保



```
#include "stdio.h"
#include <math.h>
void bar( int len )
{
    int i;
    for (i=0; i<len; i++) {
        printf("*");
    }
    printf("¥n");
    return;
}
int main()
{
    int len;
    char buf[256];
    int ch;
    printf( "len = " );
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%d¥n", &len );
    bar( len );
    ch = getchar();
    ch = getchar();
    return 0;
}
```

len をメモリエリア中に確保

変数 i をメモリエリア中に確保

データ型は 2 種類  
使っている

変数 len, buf, ch  
をメモリエリア中に確保

整数を扱う int 型  
整数は, 5, -3, 0 など

文字を扱う char 型

文字は, 1, 0, 3, -, a など  
数字(1, 0, 3 など)も文字の一種

# 例題. 平方根の計算



- 浮動小数データを読み込んで、平方根の計算と表示を行うプログラムを作る。
  - 但し、負の数の場合には、メッセージを表示すること。
  - 負の数であるかどうかによって条件分岐を行うために if 文を使う。

例) 9 のとき : 3

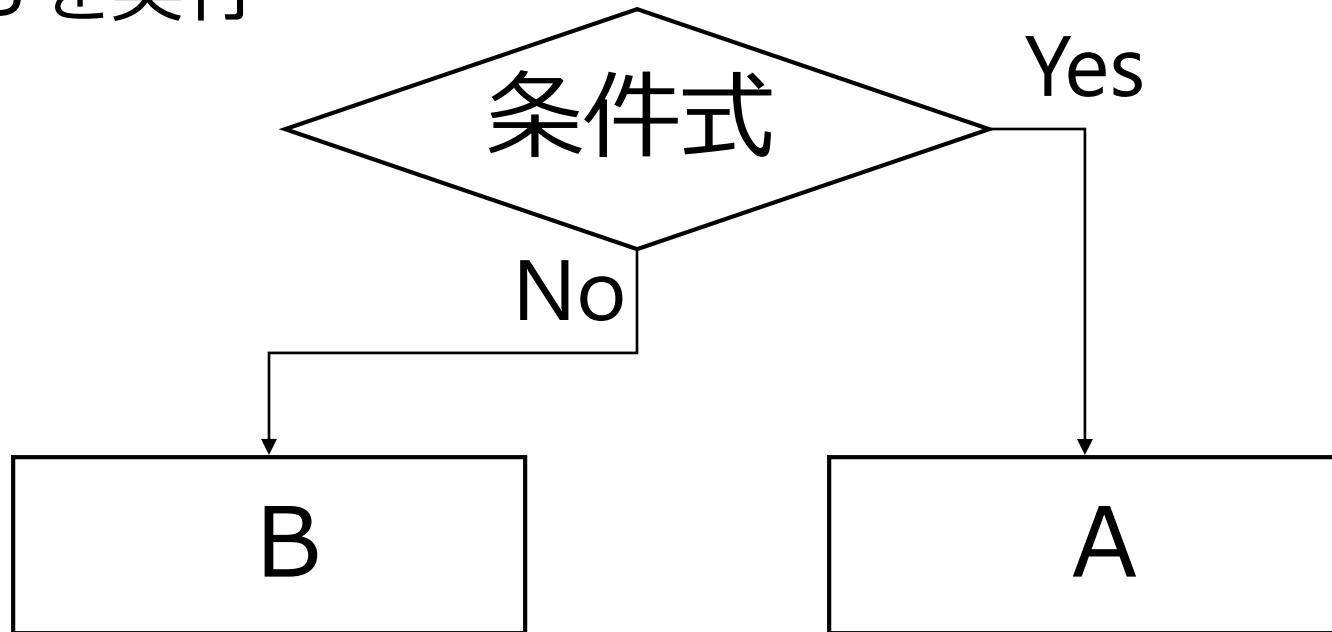
- 1 のとき : メッセージを表示



# 条件分岐とは



- 「ある条件式」が成り立てばAを、成り立たなければBを実行



## 実行結果の例

$x=9$

$\text{sqrt}(9.000000)=3.000000$

$x=-1$

**負なので計算できません**



```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    printf("x=");
    fgets( buf, 256, stdin );
    sscanf_s( buf, "%lf\n", &x );
    if ( x < 0 ) {
        printf("負なので計算できません\n");
    }
    else {
        y = sqrt(x);
        printf("sqrt(%f)=%f\n", x, y);
    }
    ch = getchar();
    ch = getchar();
    return 0;
}
```

## 条件式

条件が成り立つ場合に実行される部分

条件が成り立たない場合に実行される部分



## 実行順 ( $x < 0$ ) の場合

```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    ① printf("x=");
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%lf%n", &x );
    if ( x < 0 ) {
    ④ printf("負なので計算できません\n");
    }
    else {
        y = sqrt(x);
        printf("sqrt(%f)=%f\n", x, y);
    }
    ⑤ ch = getchar();
    ⑥ ch = getchar();
    ⑦ return 0;
}
```



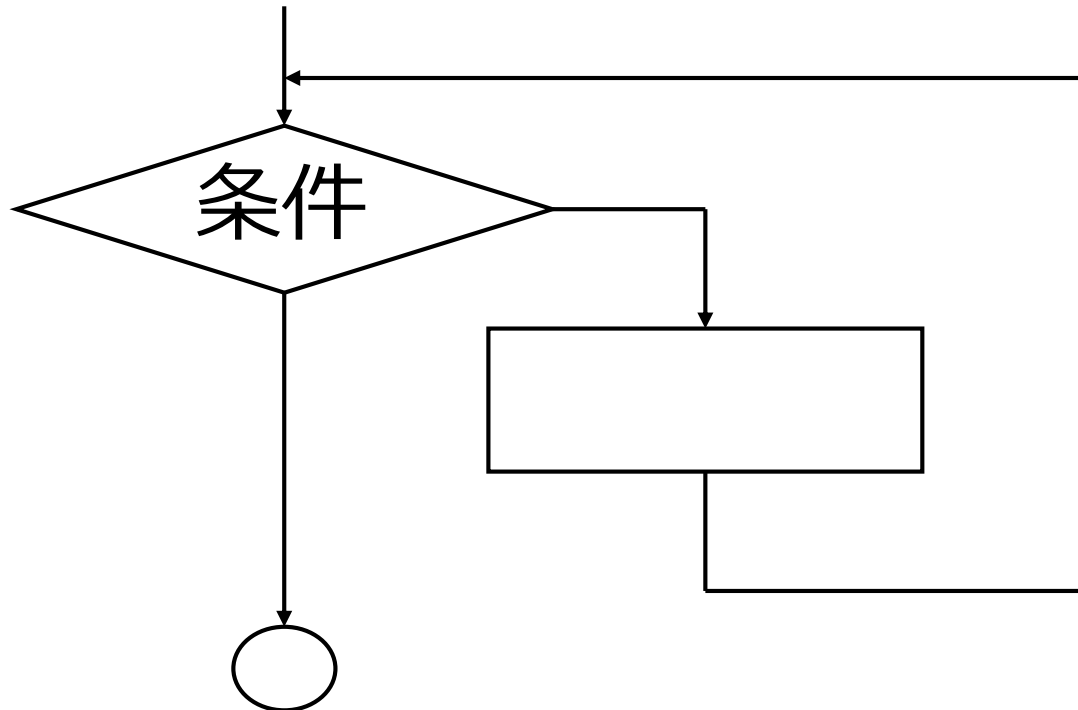
# 実行順 ( $x \geq 0$ ) の場合

```
#include "stdio.h"
#include <math.h>
int main()
{
    double x;
    double y;
    char buf[256];
    int ch;
    ① printf("x=");
    ② fgets( buf, 256, stdin );
    ③ sscanf_s( buf, "%lf%n", &x );
    if ( x < 0 ) {
        printf("負なので計算できません\n");
    }
    else {
        ④ y = sqrt(x);
        ⑤ printf("sqrt(%f)=%f\n", x, y);
    }
    ⑥ ch = getchar();
    ⑦ ch = getchar();
    ⑧ return 0;
}
```

# 繰り返しとは



- 繰り返しとは、ある条件が満たされるまで、同じことを繰り返すこと。
- 繰り返しを行うための文としてwhile文, for 文 などがある。

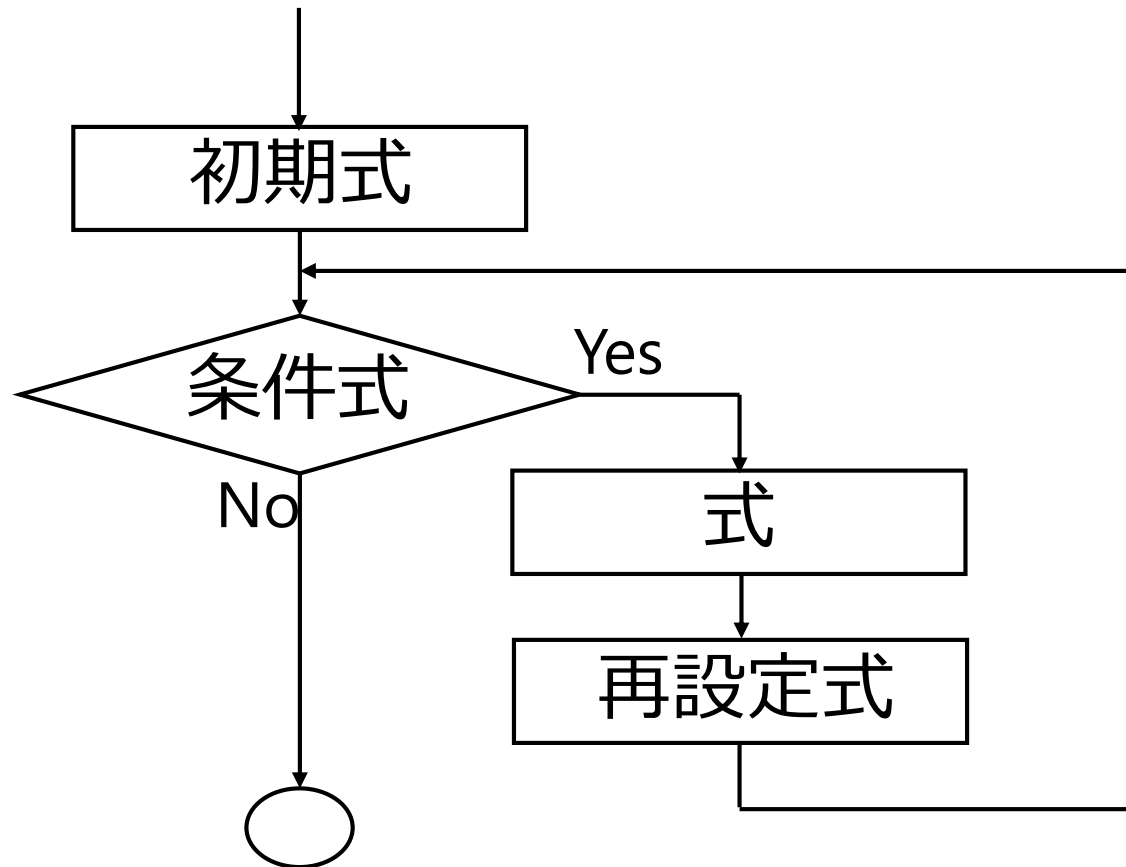


# 繰り返し



- C言語での繰り返しは次の3通り
  - while文
  - do-while文
  - for文
- ループ変数 (ループカウンタ)
  - 繰り返しの回数を数える変数
  - インクリメント 値を1増やす (i++)
  - デクリメント 値を1減らす (i--)

# for 文による繰り返し



1. まず, 「初期式」 を実行
2. 次に, 「条件式」 を実行. 条件式が成立すれば, 式と「再設定式」 を実行し, 「条件式」 に戻る





# 例題. 繰り返し計算と ファイル出力

- 次のページのプログラムの機能
- 計算の繰り返し
- キーボードからのデータ読み込み
- ファイルへの書き出し