ce-1. C プログラミング入門
（Cプログラミング応用，全１４回）

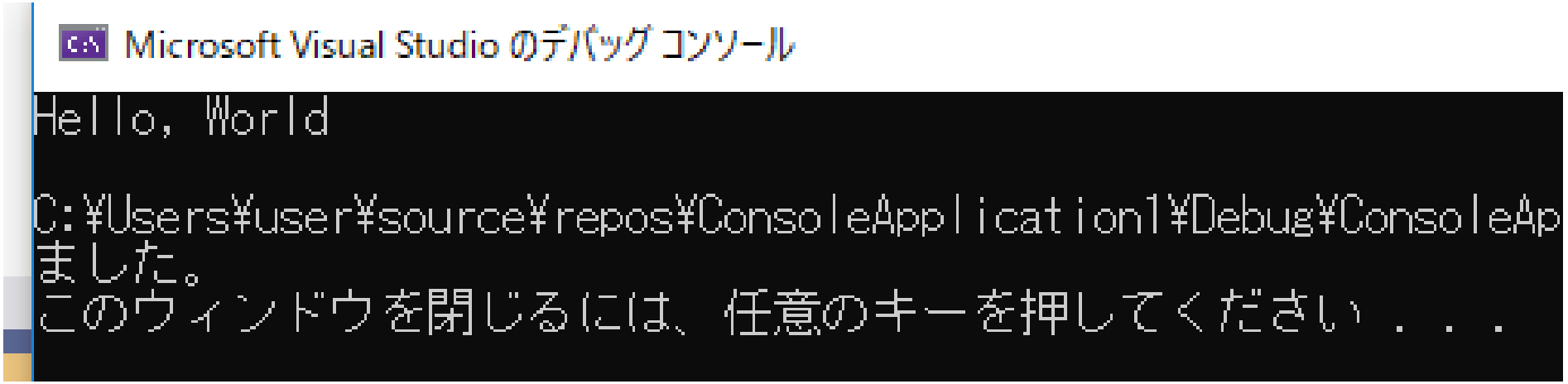
https://www.kkaneko.jp/cc/c/index.html

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
- きめ細かな操作ができる
  - メモリの操作など
- 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



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

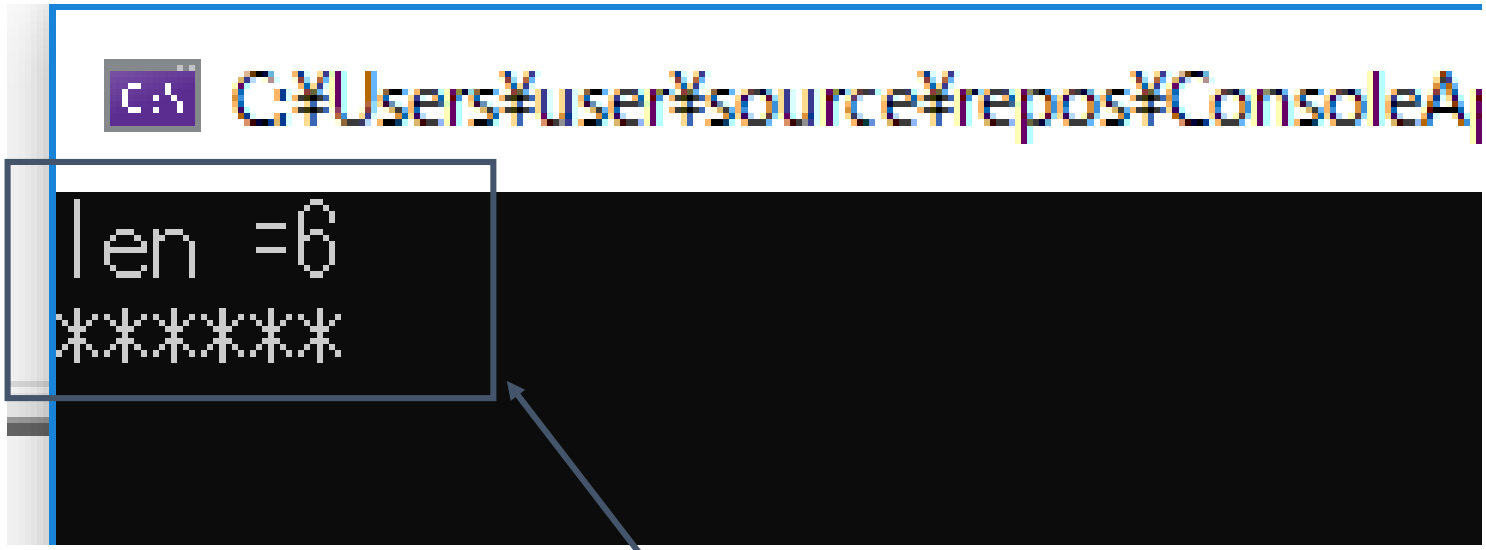
例題１．棒グラフ

- 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



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;
}
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;
}
```

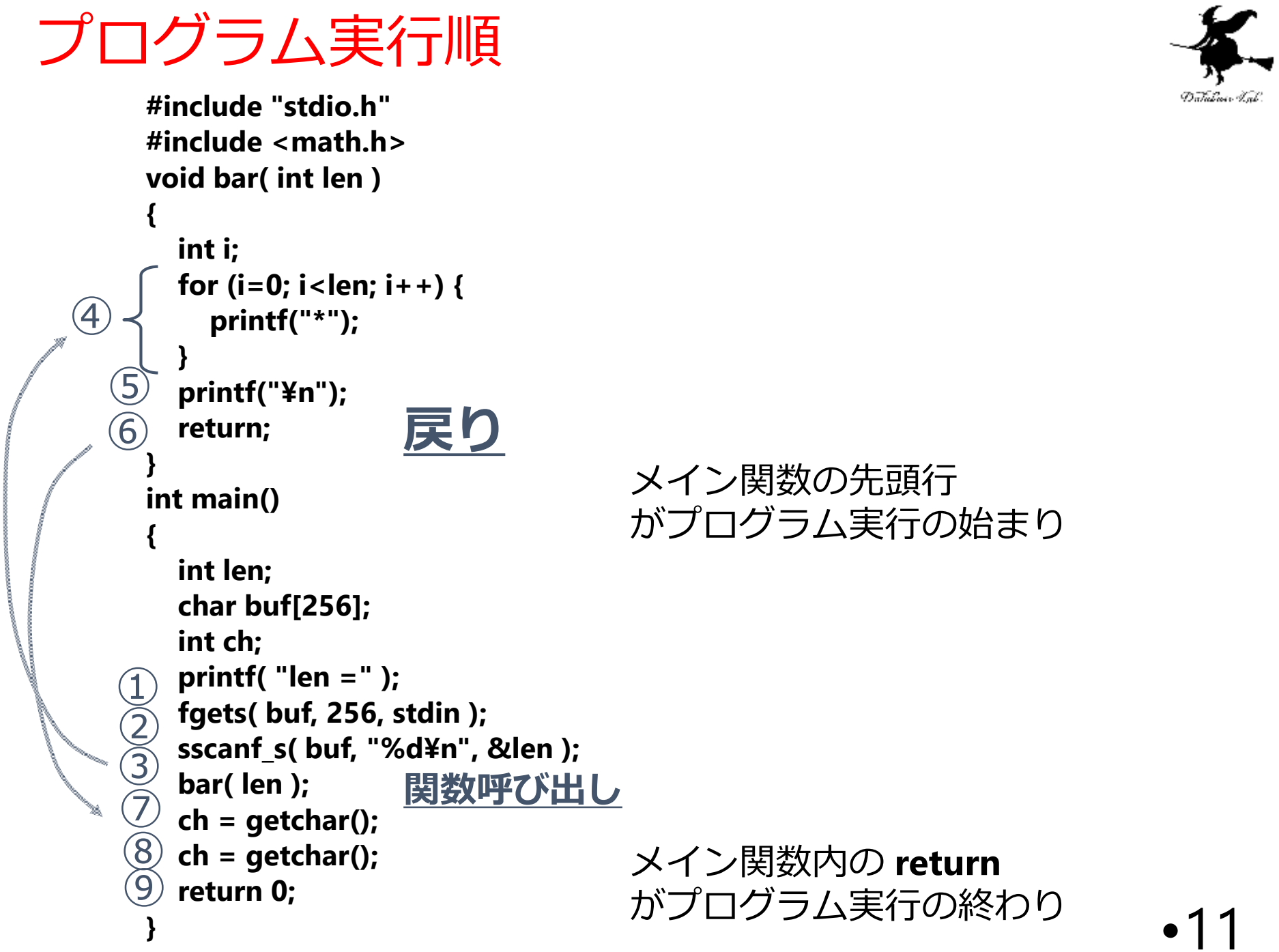
bar関数

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++) {
④       printf("*");
    }
⑤   printf("¥n");
⑥   return;
}
int main()
{
    int len;
    char buf[256];
    int ch;
①   printf( "len =" );
②   fgets( buf, 256, stdin );
③   sscanf_s( buf, "%d¥n", &len );
    bar( len );
⑦   ch = getchar();
⑧   ch = getchar();
⑨   return 0;
}
```


メイン関数の先頭行
がプログラム実行の始まり

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

```
#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;
}
```

プログラムの実行開始

メイン関数

関数呼び出し

bar(len);

bar 関数

return;

プログラムの実行終了

* printf, scanf の
呼び出しについては
図では省略

変数

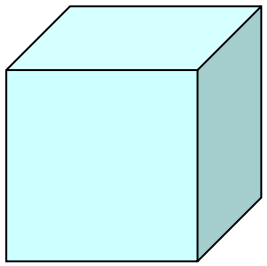
- 変数

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;
}
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

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

変数 **len, buf, ch** をメモリエリア中に確保

データ型は２種類使っている

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

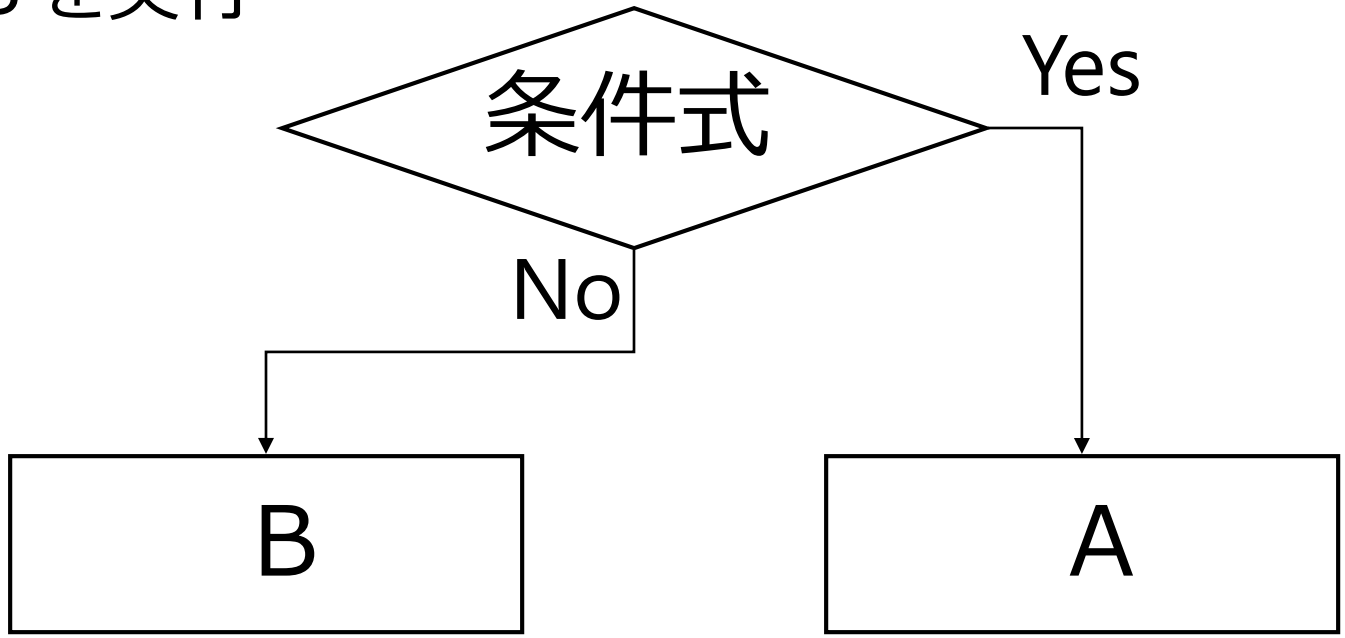
- 浮動小数データを読み込んで，平方根の計算と表示を行うプログラムを作る．
  - 但し，負の数の場合には，メッセージを表示すること．
  - 負の数であるかどうかによって条件分岐を行うために if 文を使う．

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



平方根の計算

実行結果の例

x=9

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

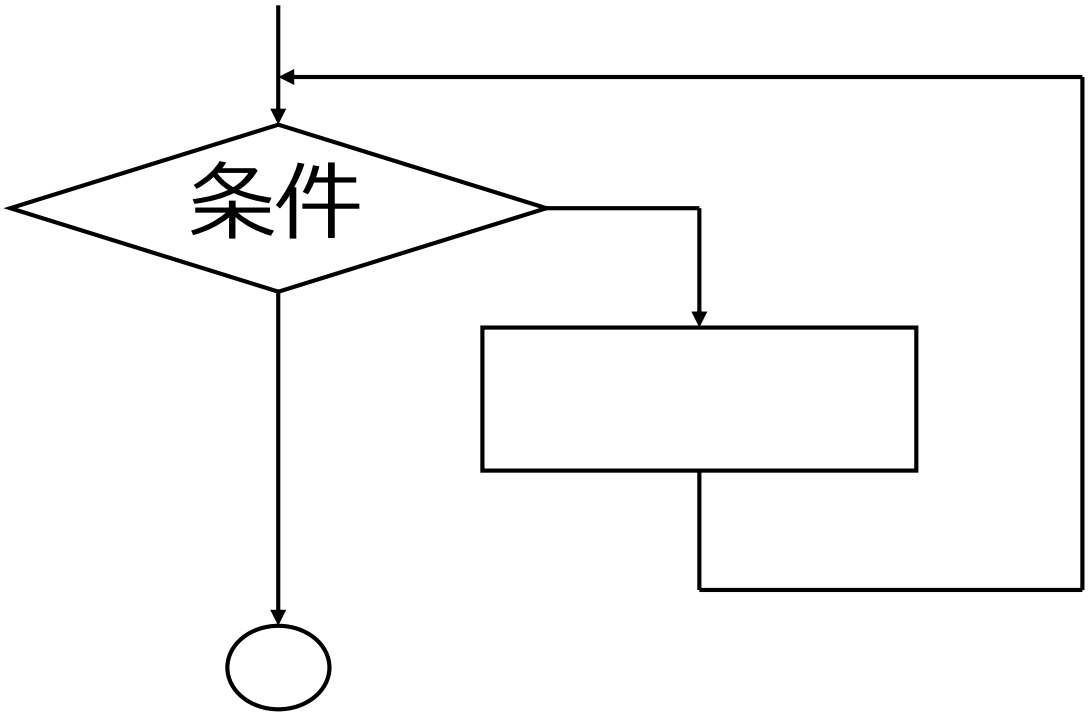
```
#include "stdio.h"
#include <math.h>
int main()
{
  double x;
  double y;
  char buf[256];
  int ch;
① printf("x=");
② fgets( buf, 256, stdin );
③ sscanf_s( buf, "%lf¥n", &x );
  if ( x < 0 ) {
④   printf("負なので計算できません¥n");
  }
  else {
    y = sqrt(x);
    printf("sqrt(%f)=%f¥n", x, y);
  }
⑤ ch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

実行順
(x≧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

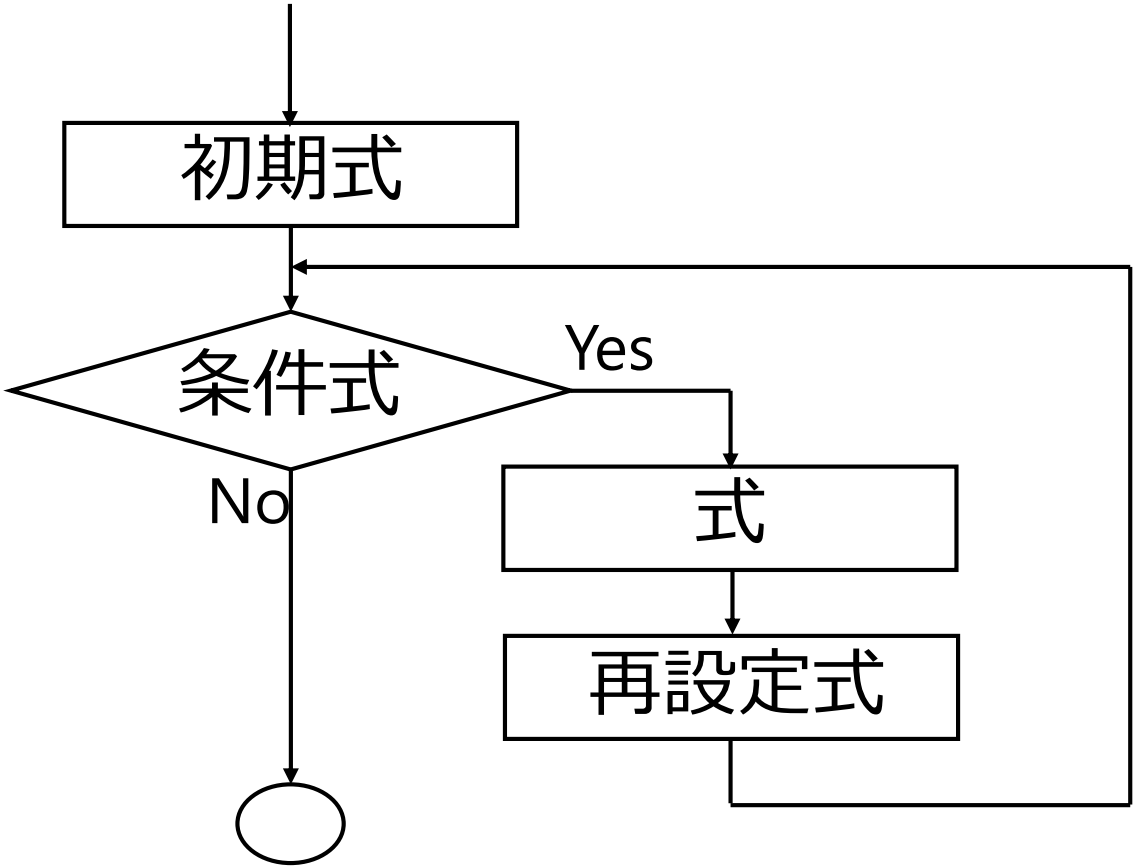
- 繰り返しとは，ある条件が満たされるまで，同じことを繰り返すこと.
- 繰り返しを行うための文としてwhile文, for 文 などがある.



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