

第7回 条件による繰り返し

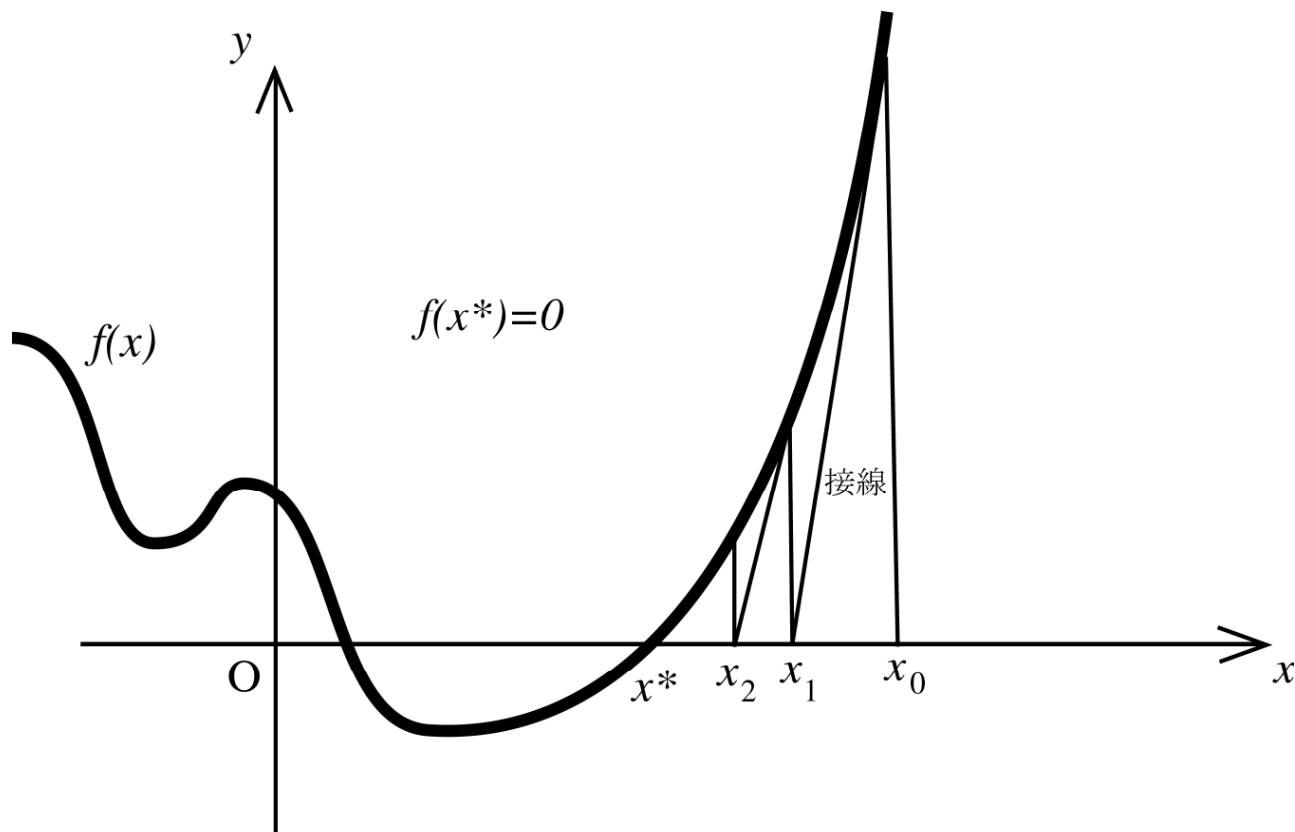


今回の目標

- アルゴリズムの基本となる制御構造（順次、分岐、反復構造）を理解する。
 - 繰り返し（反復構造、ループ）を理解する。
 - ループからの様々な終了の方法を理解する。
 - 繰り返しを用いたアルゴリズムに慣れる。
- ☆ニュートン法を用いた平方根の計算プログラムを作成する。

ニュートン法

$f(x) = 0$ の解 x^* を数値計算で求める方法

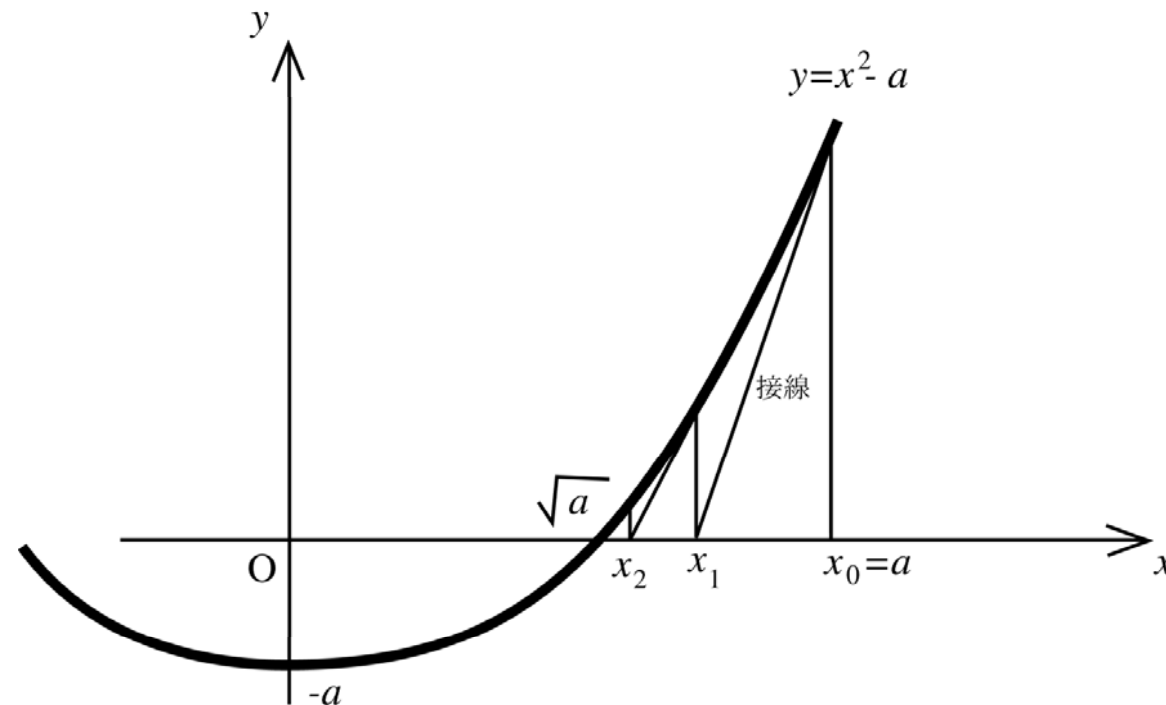


$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_0, x_1, x_2, \dots, x_\infty \rightarrow x^*$$

ニュートン法による平方根の計算

\sqrt{a} は $f(x) = x^2 - a = 0$ の解なので、



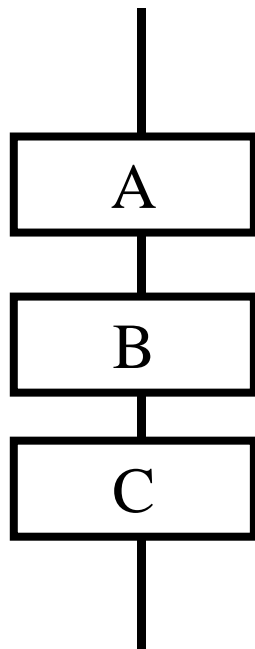
$$x_{i+1} = x_i - \frac{x_i^2 - a}{2x_i}$$

$$= \left(x_i + \frac{a}{x_i} \right) \times \frac{1}{2}$$

$$a = x_0, x_1, x_2, \dots, x_\infty \rightarrow \sqrt{a}$$

プログラムの制御構造

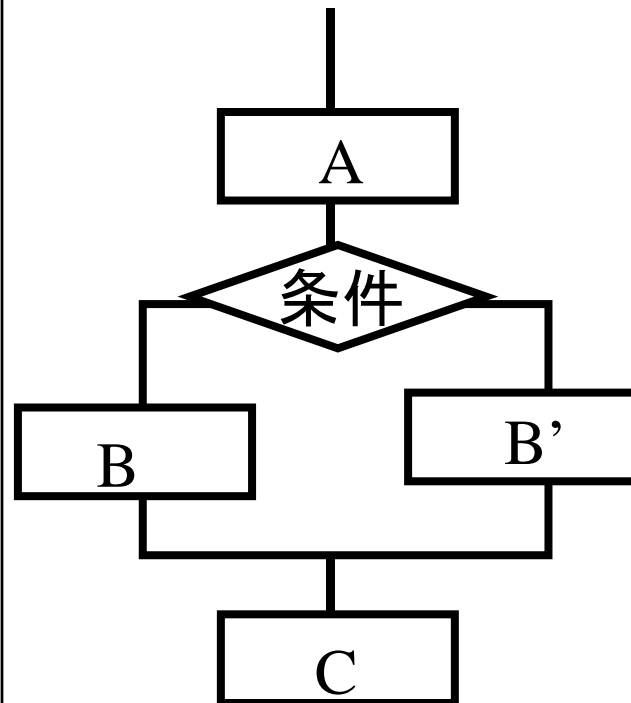
順次構造 (接続構造)



$A \rightarrow B \rightarrow C$

の順序で実行される

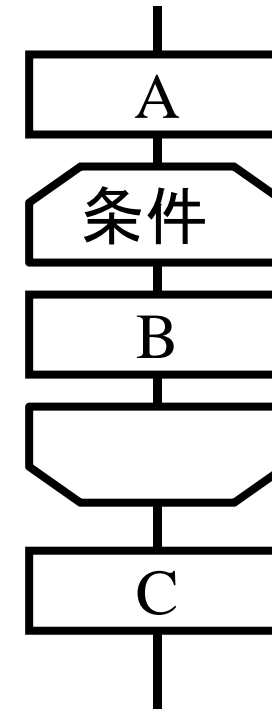
選択構造 (分岐構造)



$A \rightarrow \begin{pmatrix} B \\ B' \end{pmatrix} \rightarrow C$

の順序で実行される

反復構造 (繰り返し構造)



$A \rightarrow \underbrace{B \rightarrow B \rightarrow \cdots B}_{\text{反復}} \rightarrow C$

の順序で実行される
(反復回数は条件で制御)

基本的な制御構造は、この3種類だけ

繰り返し

- 条件(論理式)が真の間繰り返す。

主に、while文を用いると良い。

- 回数を指定して繰り返す。

主に、for文を用いると良い。(第8回で詳しく扱う。)

C言語では、主にこの2つの文を用いて繰り返しを記述する。

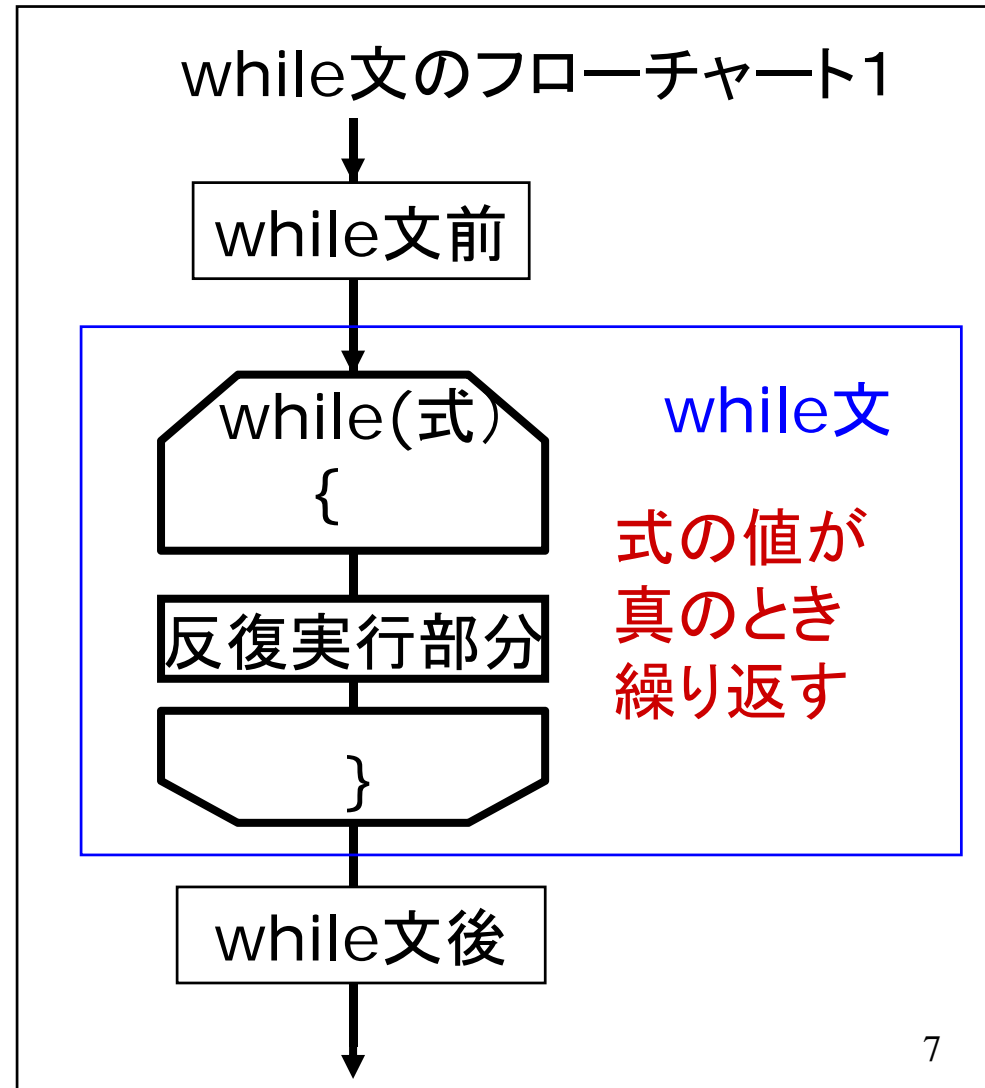
while文

式(論理式)が真である間、命令を繰り返し実行する。

書式

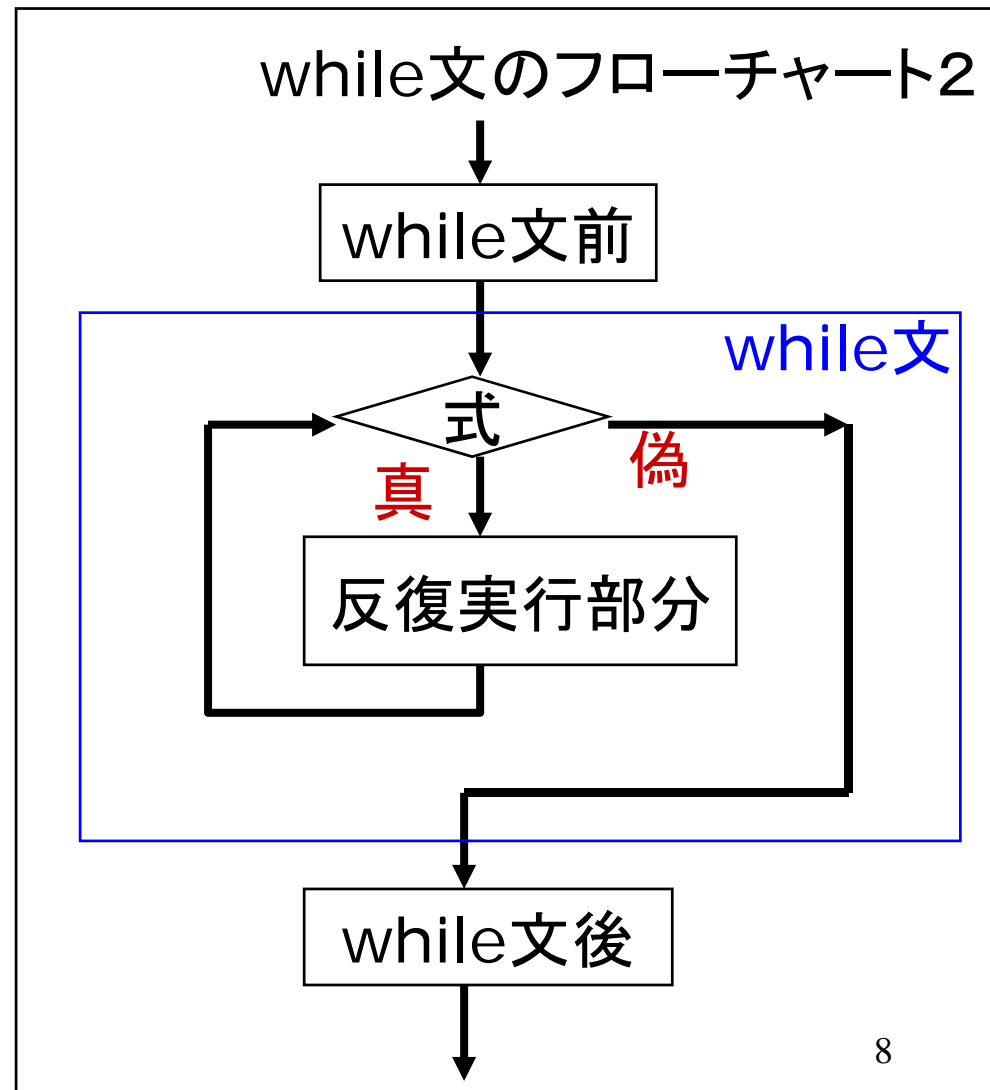
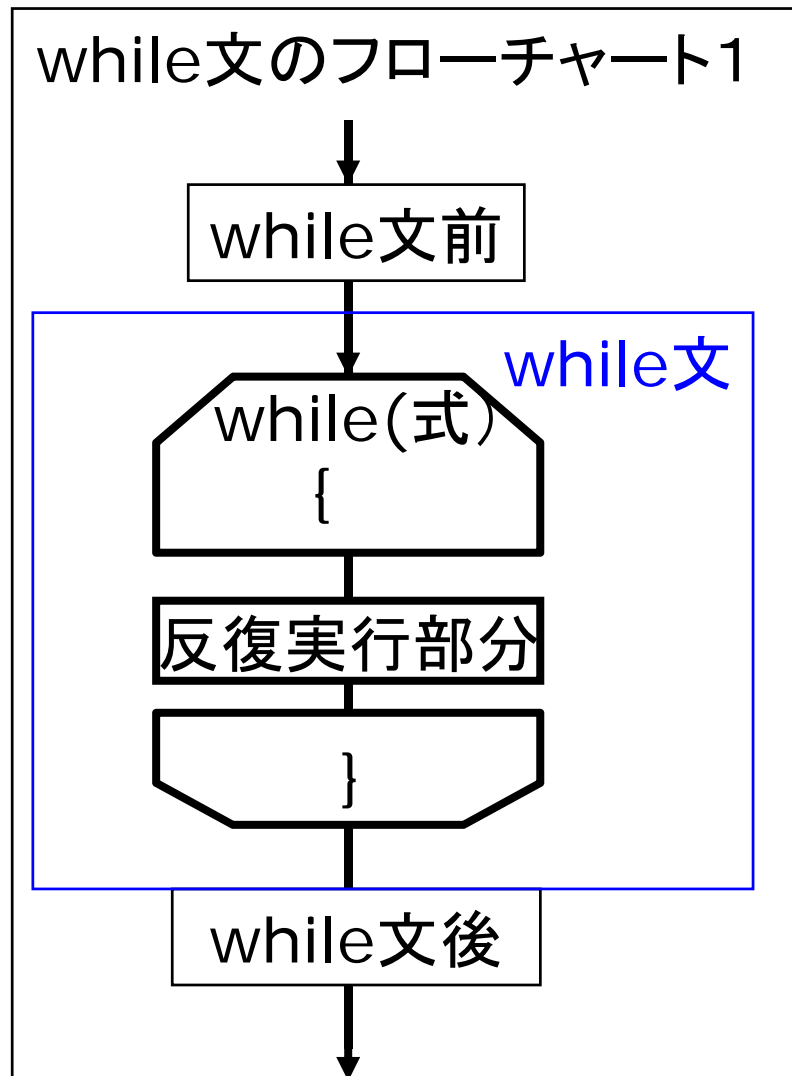
```
while(式)
{
    反復実行部分
}
```

式: 反復を続ける条件
を表す論理式



whileループのフローチャート

繰り返し文をループと呼ぶ事もある。



練習1

```
/*while 文実験1 while_test.c      コメント省略 */
#include<stdio.h>
int main()
{
    int a;
    a=1; /* 最初の一回を必ず実行するため真値を代入 */
    printf("実験開始 ¥n");
    while(a)
    {
        printf("aの値は0以外(真)です。¥n");
        printf("1(真)か0(偽)を入力して下さい。¥n");
        scanf("%d",&a);
    }
    printf("aの値が0(偽)になりました。¥n");
    printf("実験終了¥n");
    return 0;
}
```

繰り返しを回数で決めるには(while文)

ループカウンタを用いるとよい。

1. ループカウンタを整数型で用意する(宣言する)。
2. while文直前でループカウンタを0に初期化する。
3. while文の論理式を
 ループカウンタ < 望む回数
とする。
4. while文の反復実行部分最後
 (右中括弧の直前)で、
 ループカウンタをインクリメントする
 (1増やす)。

典型的な書き方

```
#define LOOPMAX 100
    .
    .
int main()
{
    int i;    /* ループカウンタ */
    .
    .
    i=0;    /* ループカウンタの初期化 */
    while( i < LOOPMAX) /* 条件判断 */
    {
        .
        .
        i++; /* ループカウンタのインクリメント */
    }
```

無限ループとその停止

終わらないプログラムの代表として、無限ループがある。
いろいろなプログラムを作る上で、
無限ループを知っていなければならない。

無限ループの書き方

```
while(1)
{
}

```

プログラムが終わらないときには、
プログラムを実行しているkterm上で、
コントロールキーを押しながら、cキーを
押す。(C-c)

それでも
終わらないときは、
教員に相談

練習2

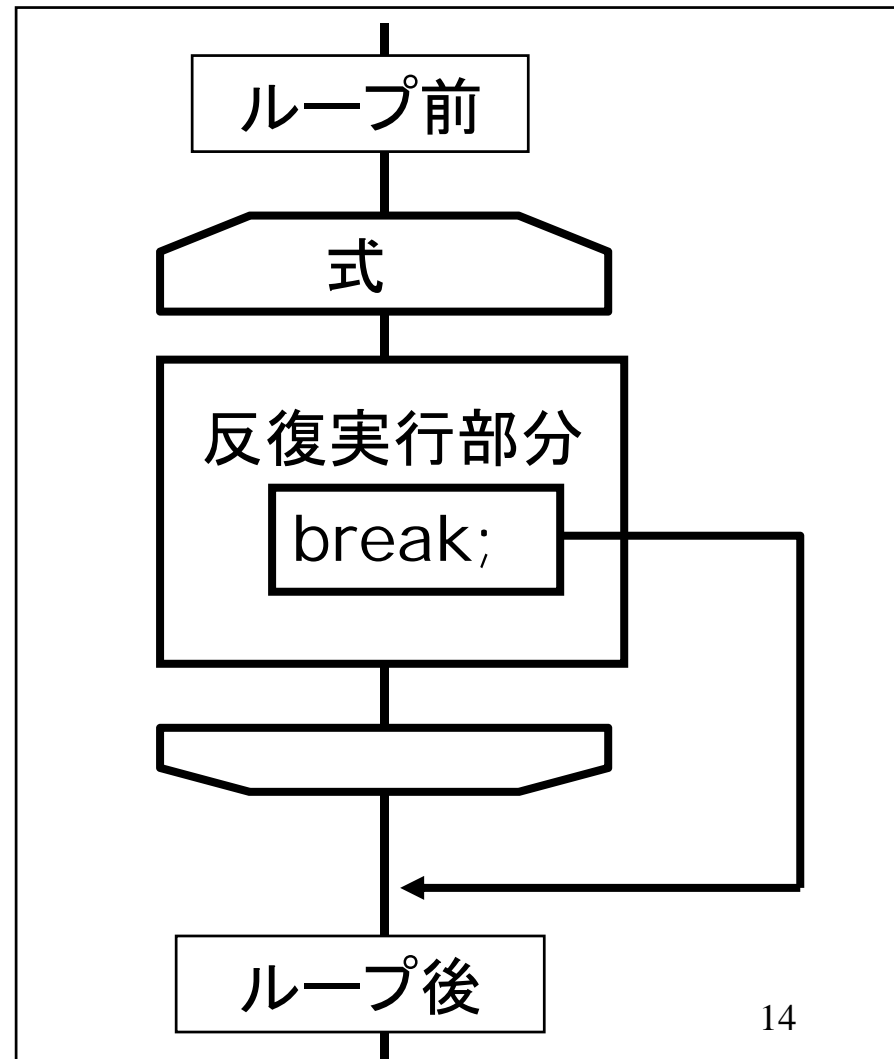
```
/* infty_loop.c 無限ループ実験(コメント省略) */  
#include <stdio.h>  
int main()  
{  
    printf("無限ループ実験開始 ¥n");  
    while(1)  
    {  
        printf("*¥n");  
        printf("**¥n");  
        printf("***¥n");  
        printf("****¥n");  
        printf("*****¥n");  
        printf("*****¥n");  
    }  
    printf("常に実行されない。¥n");  
    return 0;  
}
```

break文

反復実行部分内で用い、breakに出会うと繰り返し文が終了する。(次の実行は、ループを閉じる右中括弧直後から)

書式

```
while( 式)  
{  
    反復実行部分内のどこか  
    (break;)  
}
```

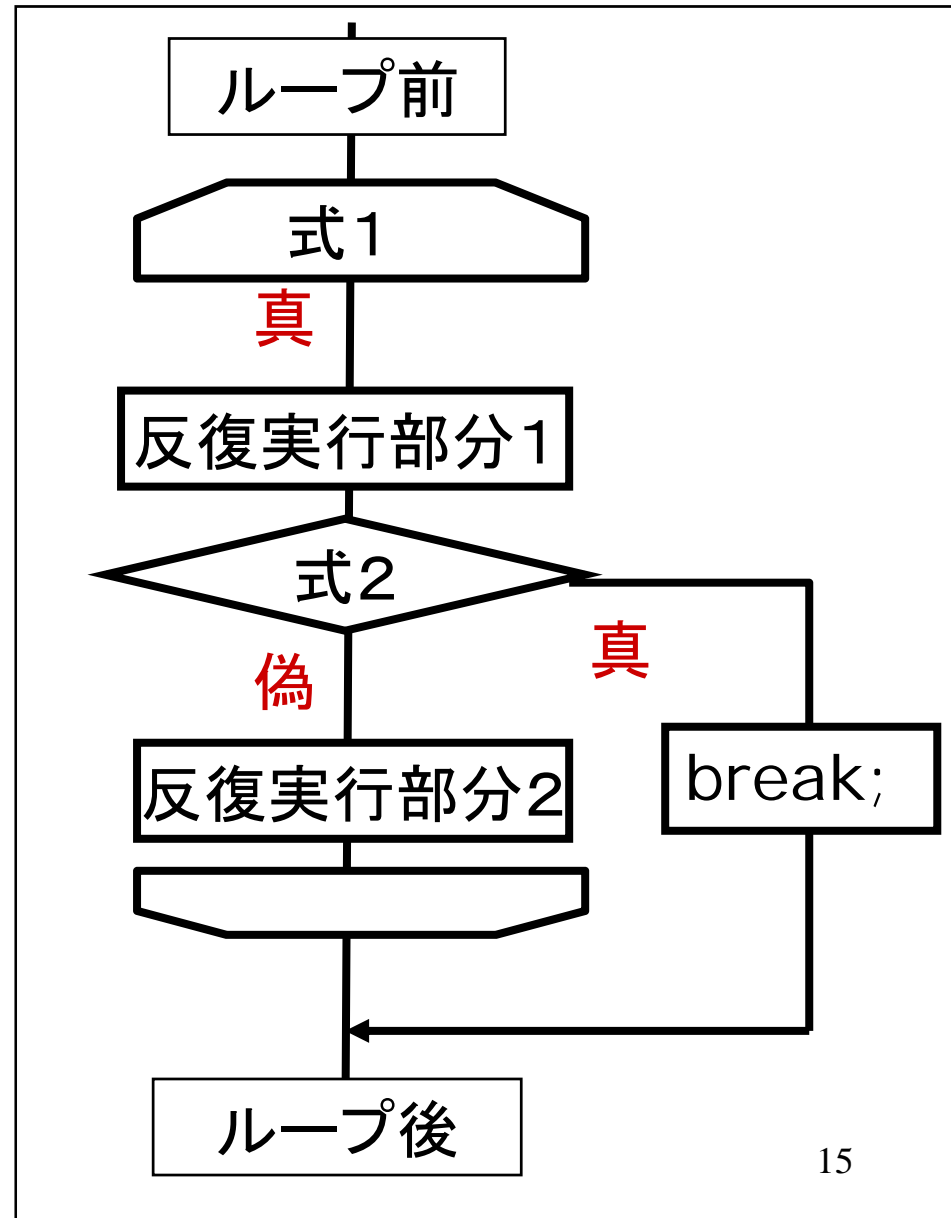


break文の典型的な使い方

典型的な使い方

```
while( 式1)
{
    反復実行部分1
    if(式2)
    {
        break;
    }
    反復実行部分2
}
```

2つの条件(式)でループが終了するので注意して使うこと。

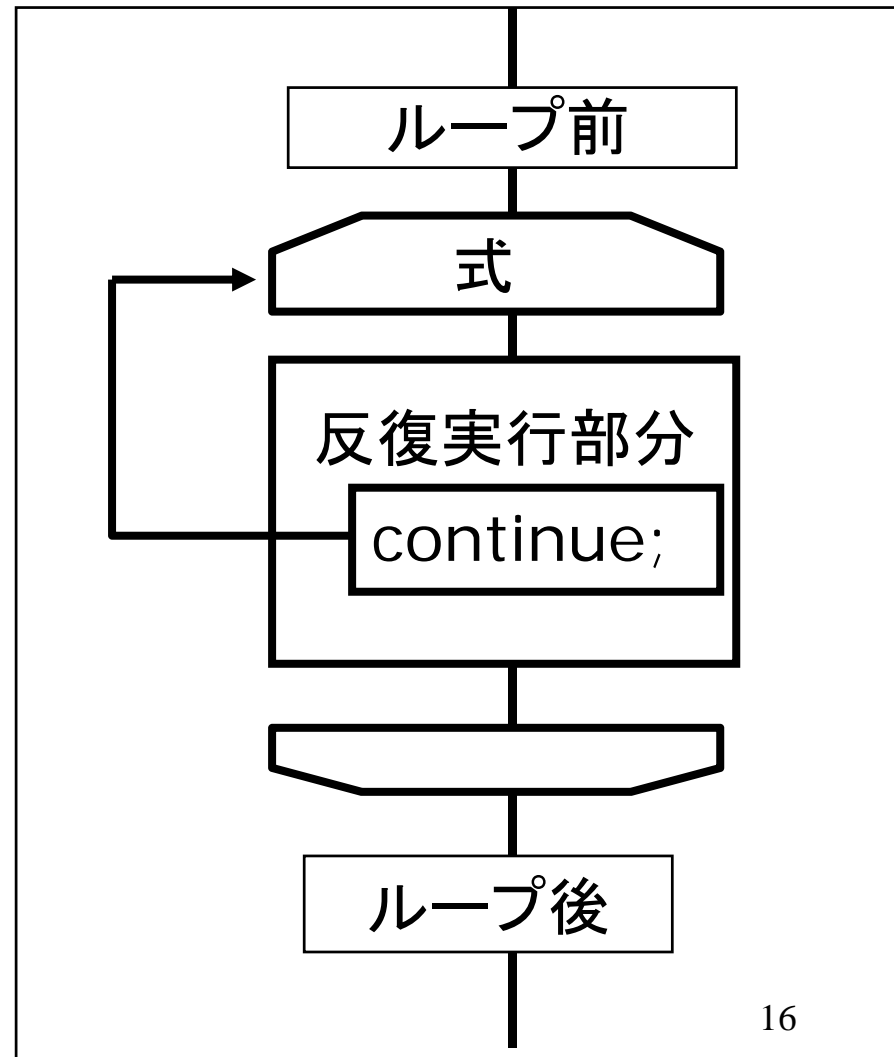


continue文

反復実行部分内で用い、continue文に出会うと
次の繰り返しを実行する。(for文の場合、式3は実行される。)

書式

```
while(式)
{
    反復実行部分内のどこか
    (continue;)
}
```

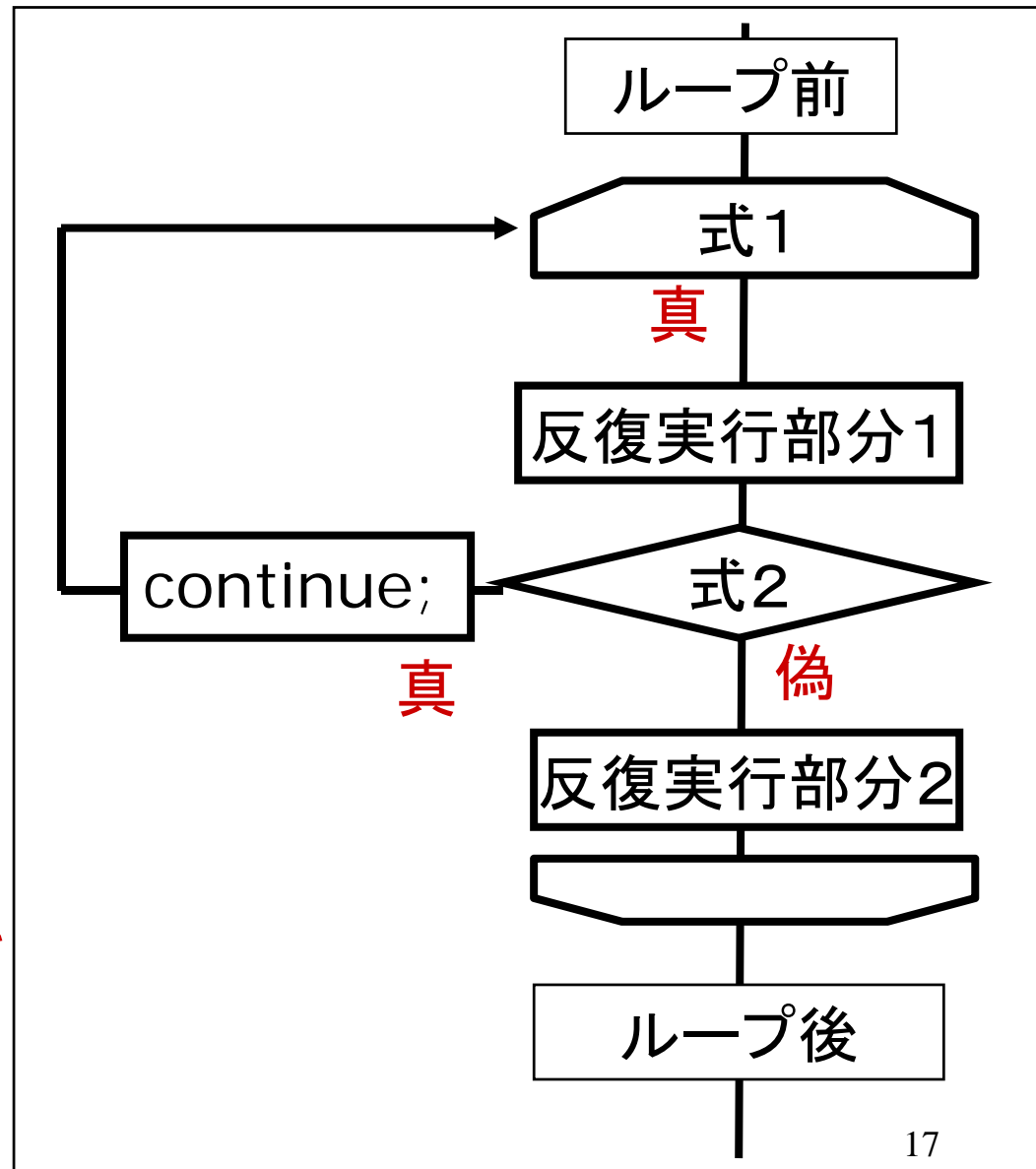


continue文の典型的な使い方

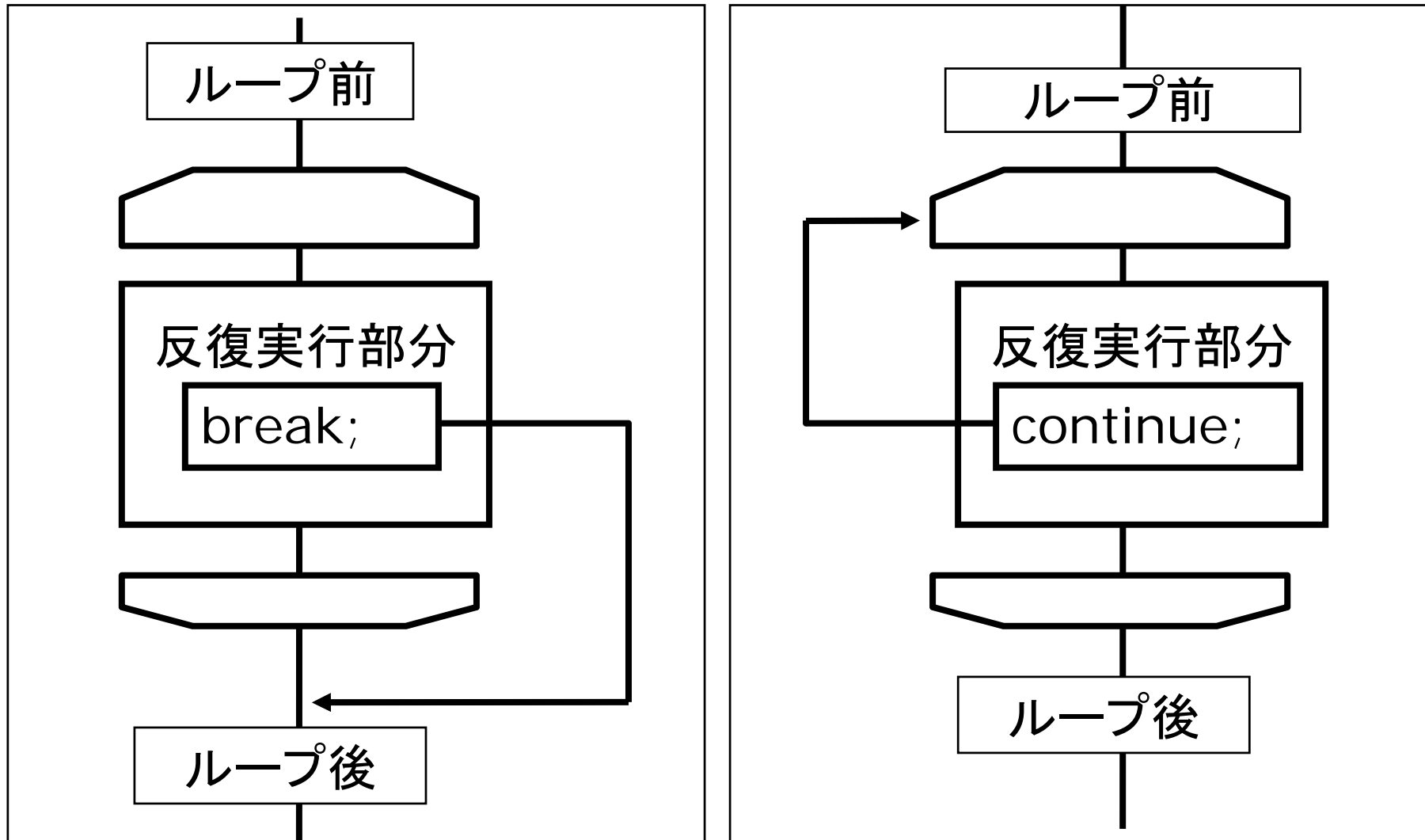
典型的な使い方

```
while(式1)
{
    反復実行部分1
    if(式2)
    {
        continue;
    }
    反復実行部分2
}
```

ループ中に反復実行文2を
実行するか選択できる。

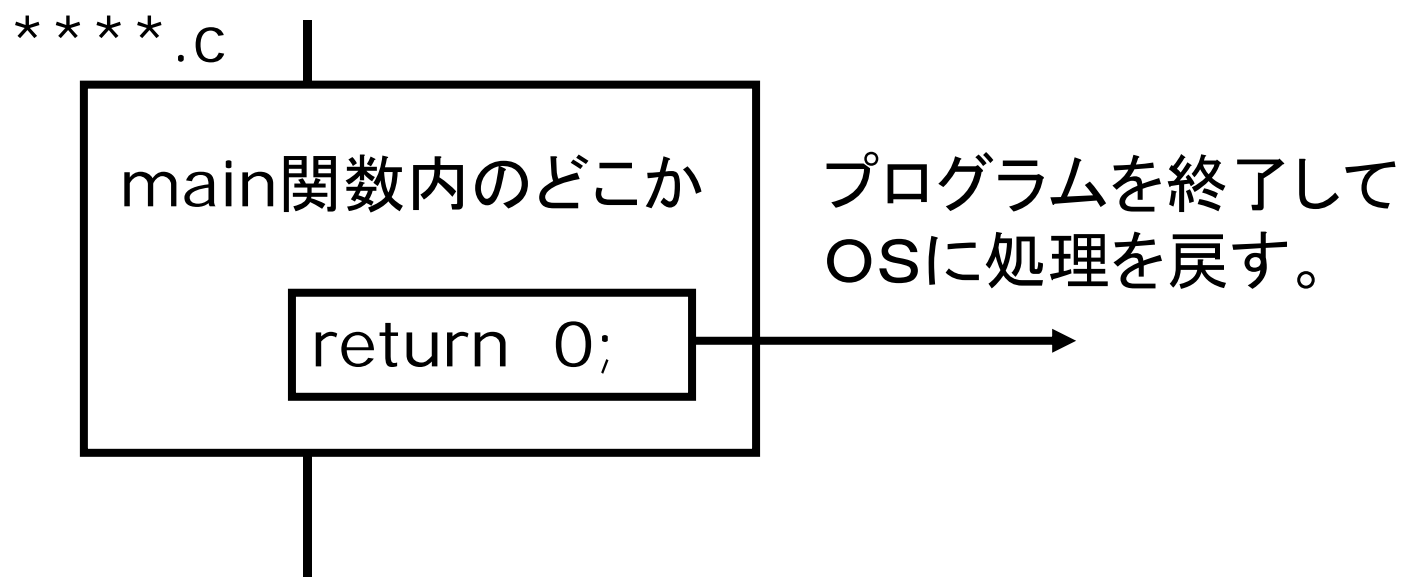


continue文とbreak文



return文

main関数で、return文を実行すると、
プログラムが終了する。
(詳しくは、第9回の関数で説明。)



最後でなくても、プログラムを終了させられる。
(エラーチェック等で使うと良い。)

平方根を求めるプログラム

/*

作成日:yyyy/mm/dd

作成者:本荘太郎

学籍番号:B0zB0xx

ソースファイル:mysql.c

実行ファイル:mysql

説明:ニュートン法を用いて平方根を求めるプログラム。

近似値の階差の絶対値がEPS(1.0e-5)より小さいときに
終了する。(繰り返し回数がLOOPMAX(1000)回に達しとき
にも終了する。)

数学関数を用いるので、-lmのコンパイルオプションが必要。

入力:標準入力から1つの実数を入力する。(正、0、負いずれも可)

出力:入力の平方根が実数の範囲で存在するとき、

標準出力にその平方根を出力する。

入力の平方根が実数でないとき、

標準出力にエラーメッセージを出力する。

*/

/*

次のページに続く

*/

```

#include <stdio.h>
#include<math.h>
/*マクロ定義*/
#define      EPS    (1.0e-5)      /*微小量、収束条件 */
#define      LOOPMAX    1000    /*  繰り返しの上限値*/

int  main()
{
    /*      変数宣言      */
    double  input;      /*  入力される実数,ニュートン法の初期値*/
    double  old_x;      /*  漸化式の右辺で用いるx*/
    double  new_x;      /*  漸化式の左辺で用いる変数x*/
    double  sa;          /*  階差の絶対値*/
    int      kaisuu;      /*  繰り返し回数*/
    /*      次ページへ続く  */

```

```
/* 入力処理 */
printf("平方根を求めます。¥n");
printf("正の実数を入力して下さい。¥n");
scanf("%lf",&input);

/* 入力値チェック */
if(input == 0.0)
{
    /* input=0.0のときは、明らかに0が平方根 */
    printf("%6.2f の平方根は%6.2fです。¥n",input,0.0);
    return 0;
}
else if(input < 0.0)
{
    /* inputが負のとき */
    printf("負の数なので、実数の平方根はありません。¥n");
    return -1;
}
/* これ以降では、inputは正の実数 */
/* 次のページに続く */
```

```
/*ニュートン法の初期設定*/
old_x=input; /*ニュートン法の初期値をinputに設定*/
new_x=0.0;
sa=fabs(new_x-old_x);
kaisuu=0;
/*繰り返し処理*/
while(sa < EPS)/*階差がEPSより小さい時終了する*/
{
    printf("x%d = %15.8f ¥n",kaisuu,old_x);/*途中表示*/
    new_x=(old_x+input/old_x)/2.0; /*漸化式*/
    sa=fabs(new_x-old_x); /*階差の絶対値*/

    /* 次の繰り返しのための処理 */
    old_x=new_x;
    kaisuu++;
    if(kaisuu>=LOOPMAX)
    { /*繰り返し回数の上限を超えたので終了する*/
        break;
    }
}
/*次のページに続く*/
```

```
/* 続き */  
/* 出力処理/  
printf("%6.2f の平方根は%15.8fです。¥n",input,new_x);  
return 0;  
}
```


実行例1

```
$. /mysqrt
```

平方根を求めます。

正の実数を入力して下さい。

2.0

x0= 2.000000000

x1= 1.500000000

x2= 1.466666667

x3= 1.41421569

x4= 1.41421356

2.00の平方根は 1.41421356です。

\$

実行例2

```
$/mysqrt  
平方根を求めます。  
正の実数を入力して下さい。  
0.0  
    0.00の平方根は    0.00です。  
$
```

実行例3

```
$/mysqrt  
平方根を求めます。  
正の実数を入力して下さい。  
-1.0  
負の数なので、実数の平方根はありません。  
$
```