第6回　繰り返しI
（条件による繰り返し）

1

今回の目標

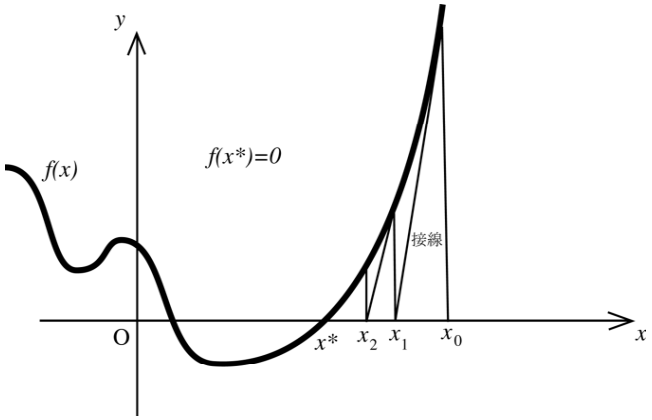
- アルゴリズムの基本となる制御構造（順次、分岐、反復構造）を理解する。
- 繰り返し（反復構造、ループ）を理解する。
- ループからの様々な終了の方法を理解する。
- 繰り返しを用いたアルゴリズムに慣れる。

☆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。

2

ニュートン法

$f(x)=0$ の解 x^* を数値計算で求める方法

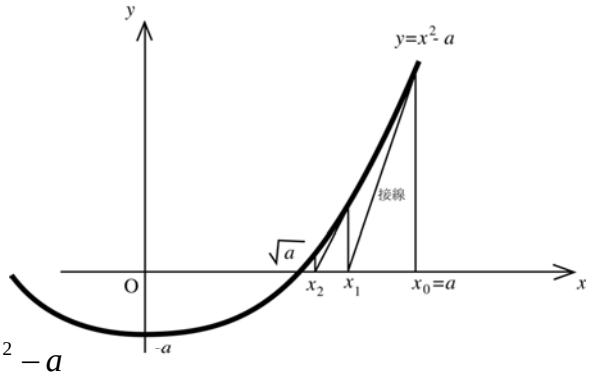


$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad x_0, x_1, x_2 \ldots, x_\infty \rightarrow x^*$$

3

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、



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

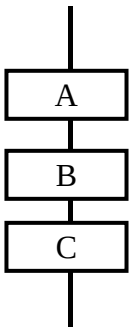
$$= \left(x_i + \frac{a}{x_i} \right) \times \frac{1}{2}$$

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

4

プログラムの制御構造

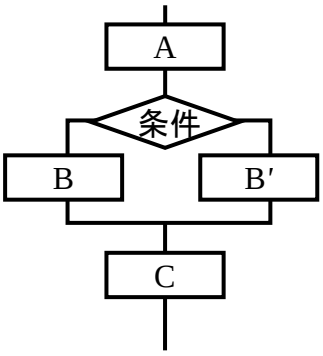
順次構造
（連接構造）



$A \to B \to C$

の順序で実行される

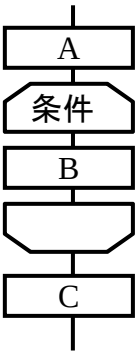
選択構造
（分岐構造）



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

の順序で実行される

反復構造
（繰り返し構造）



$A \to \underbrace{B \to B \to \cdots B} \to C$

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

5

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。
（今回、繰り返しⅠで扱う。）

- 回数を指定して繰り返す。

主に、for文を用いると良い。
（次回の繰り返しⅡ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。

6

while文

条件式（論理式）が真である間、命令を繰り返し実行する。

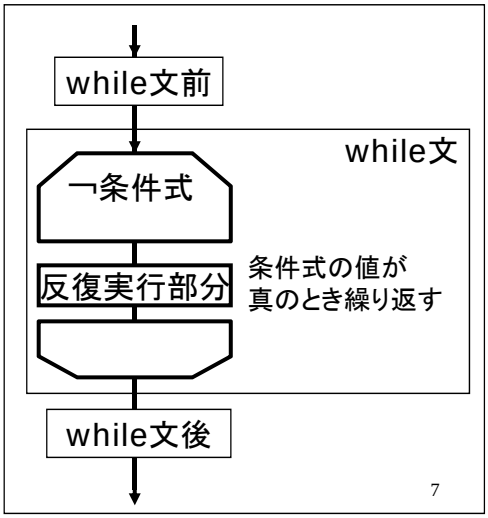
書式

```
while(条件式)
    {
        反復実行部分
    }
```

条件式:
反復を**続ける**条件
を表す論理式
（偽になったら反復終了）

while文は、
反復条件で繰り返し
を制御する。

while文のフローチャート

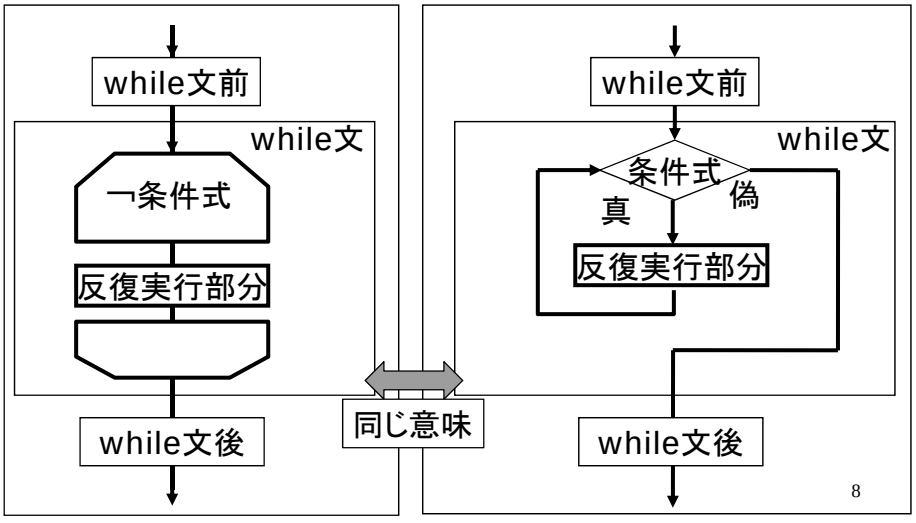


7

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。

while文のフローチャート　　省略しない書き方



8

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（整数型の変数）を用意する（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）。

指定した回数だけ繰り返し（while文）

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*/
          }
```

11

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

12

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;
}
```

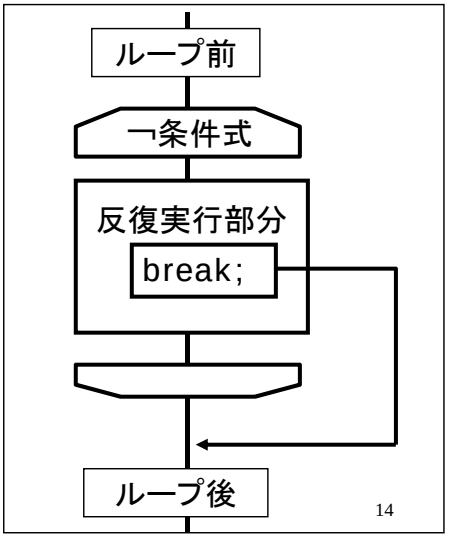
13

break文

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

書式

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



14

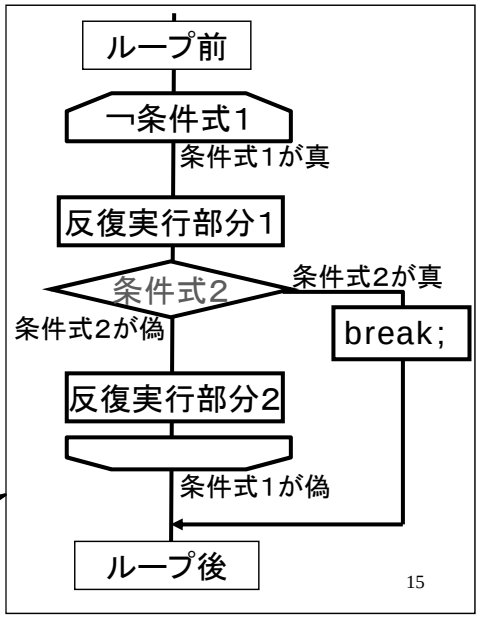
break文の典型的な使い方

典型的な使い方

```
while(条件式1)
    {
        反復実行部分1
        if(条件式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。

式2は、終了条件を意味する。



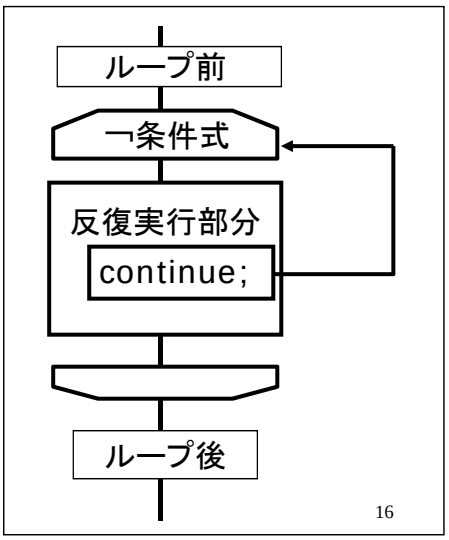
15

continue文

反復実行部分内でcontinue文に出会うと
条件式の判断の箇所まで戻る（次の繰り返し処理を実行する）。

書式

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



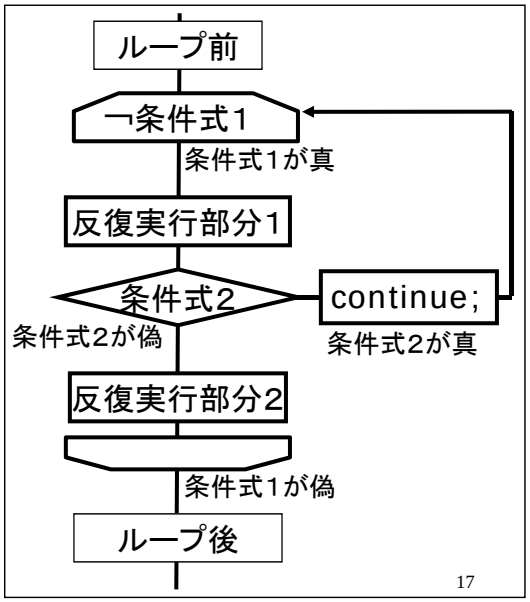
16

continue文の典型的な使い方

典型的な使い方

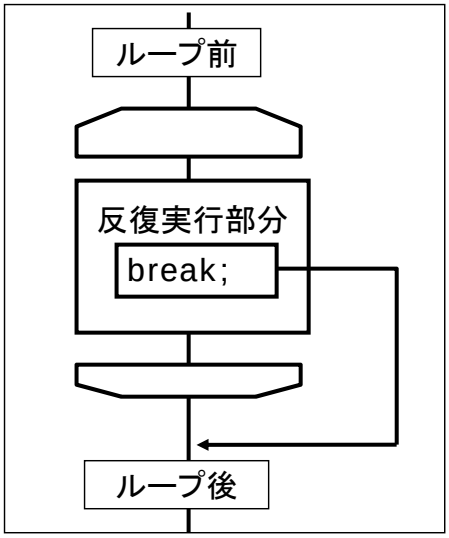
```
while(条件式1)
    {
        反復実行部分1
        if(条件式2)
            {
                continue;
            }
        反復実行部分2
    }
```

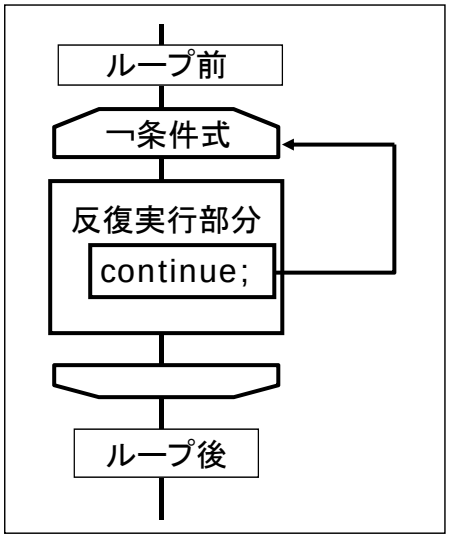
ループ中に反復実行文2を実行するか選択できる。



17

break文とcontinue文

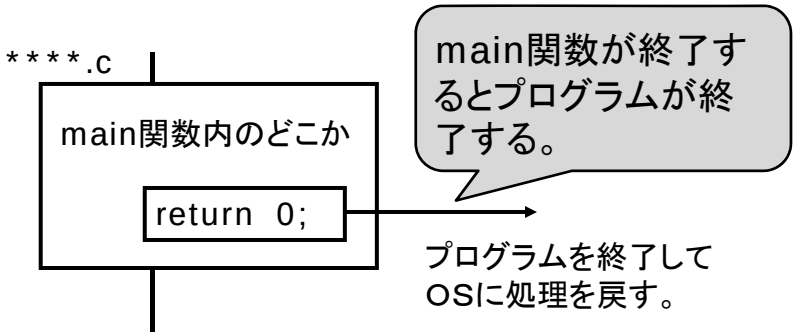




18

return文

return文は関数を終了させる。
main関数内のreturn文はプログラムが終了させる。
（詳しくは、第9～11回の関数Ⅰ、Ⅱ、Ⅲ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。）

19

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:B00B0xx
        ソースファイル:mysqrt.c
        実行ファイル:mysqrt
        説明:ニュートン法を用いて平方根を求めるプログラム。
              求められた平方根の値の二乗と、入力された値の差の絶対値が
              EPS(1.0e-5)より小さくなるまで繰り返しを行う。
              （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つの実数値を入力する。（正、0、負いずれも可）
        出力:入力された実数値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の近似値を出力する。
             計算の途中経過についても標準出力に出力する。
             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          */
```

20

```
/* 前ページからの続き */

#include <stdio.h>
#include<math.h>
/*マクロ定義*/
#define        EPS    (1.0e-5)     /*微小量、ニュートン法の収束条件 */
#define        LOOPMAX     1000      /*    繰り返しの最大回数*/

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  approx;         /*  平方根の近似値 */
        double  error;          /*  平方根の近似値の二乗と、
                                    入力された値の差の絶対値*/
        int     kaisuu;         /*繰り返し回数*/

/*      次のページに続く        */
```



```
/* 前ページからの続き */
        /* 平方根を求めるべき実数値の入力 */
        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);

        /*  入力値が正しい範囲であるかチェック  */
        if(input ==0.0)
          {     /*input=0.0のときは、明らかに0が平方根であるので*/
                /*ニュートン法を利用しなくとも良い*/
                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        */
```

```
/* 前ページからの続き */
        /*ニュートン法の初期設定*/
        approx = input;         /*ニュートン法の初期値を入力値に設定*/
        error = fabs(approx*approx - input);
        kaisuu = 0;
        /* ニュートン法の繰り返し処理 */
        while(error > EPS) /* 差がEPSより大きい間は繰り返す*/
           {
                if(kaisuu>=LOOPMAX)
                   {
                        break; /* 繰り返し回数の上限を超えたので終了 */
                   }
                /* ニュートン法の途中経過の表示 */
                printf("x%d = %15.8f ¥n", kaisuu, approx);
                 /* ニュートン法の漸化式の計算 */
                approx = ( approx + (input/approx) ) / 2.0;
                error = fabs(approx*approx - input);

                /* 次の繰り返し処理のための準備 */
                kaisuu++;
           }
/*      次のページに続く          */
```



```
/* 前ページからの続き */

        /* 計算結果の出力 */
        printf("%6.2f の平方根は%15.8fです。¥n", input, approx);
        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=       2.00000000
x1=       1.50000000
x2=       1.46666667
x3=       1.41421569
2.00の平方根は          1.41421356です。
$
```

25

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。
$
```

26