第9回関数1
（処理の分解）

1

今回の目標

- C言語における関数を理解する。
- 仮引数と実引数の役割について理解する。
- 戻り値について理解する。
- プロトタイプ宣言を理解する。
- 複数の仮引数を持つ関数を理解する。

☆階乗を求める関数を利用して、組み合わせの数を求めるプログラムを作成する

2

組み合わせの数を求める公式

$$ {}_nC_m = \frac{n!}{m! \times (n-m)!} $$

3

関数の定義

returnの後の式(変数や定数）と同じ型

一種の入力、仮引数という。
仮引数は変数の一種。
複数の場合もある。

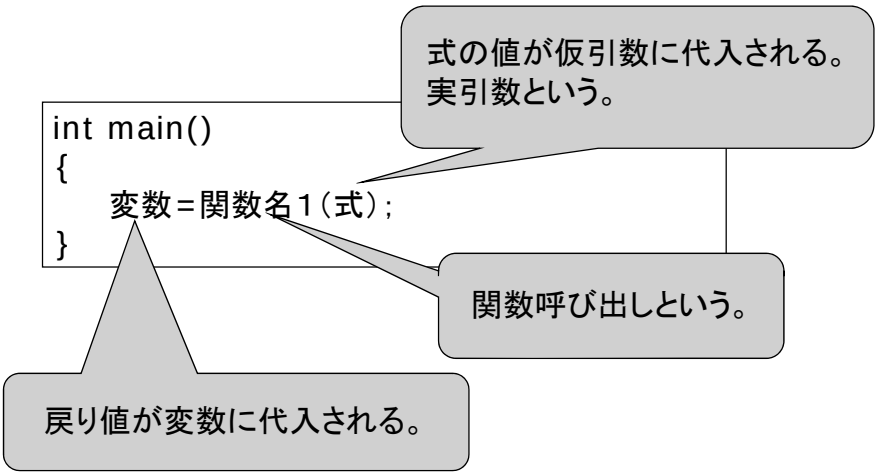
書式:

```
戻り値の型　　関数名（仮引数の宣言付きリスト）
{
        関数の本体
        return　式;
}
```

一種の出力、戻り値という。
もちろん、
定数や変数であっても良い。

4

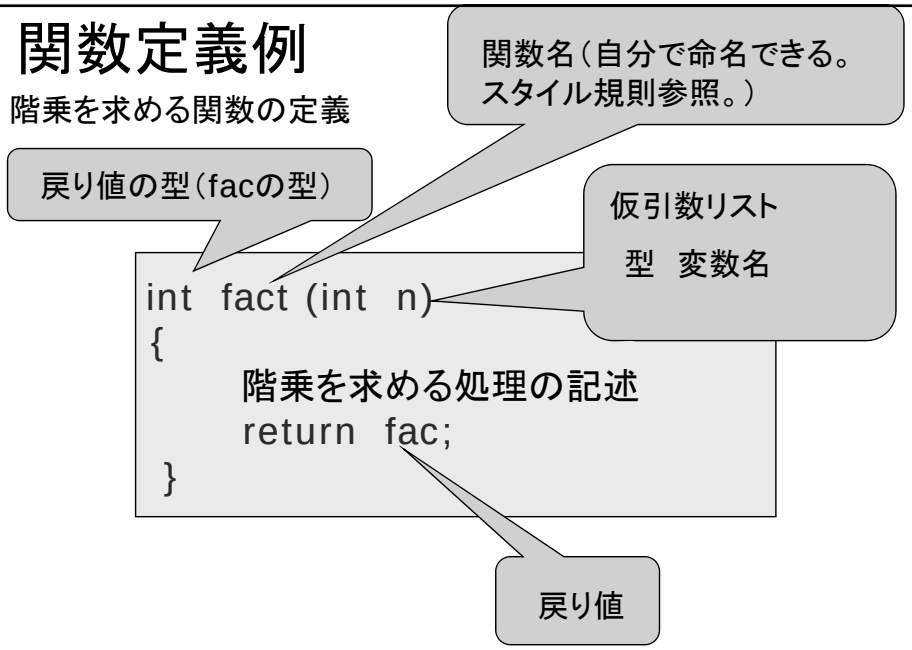
関数の典型的な利用例（関数の呼び出し）



5

関数定義例

階階乗を求める関数の定義

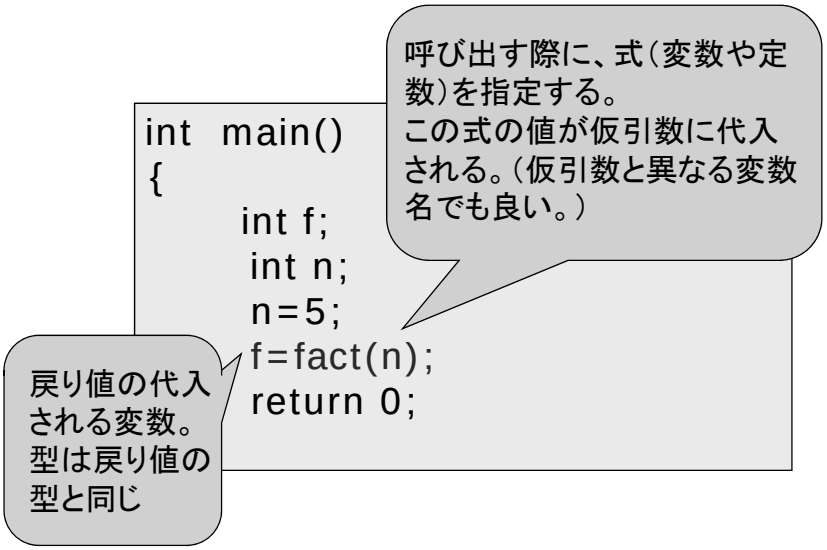


注意：メイン関数以外はプロトタイプ宣言を行うこと。

6

関数利用例

main内での関数呼び出し利用



7

mainという関数

mainというのも関数の一つ。
Cではプログラムは関数の集まりで作られる。

```
int main()
{
      関数の本体
      return 0;
 }
```

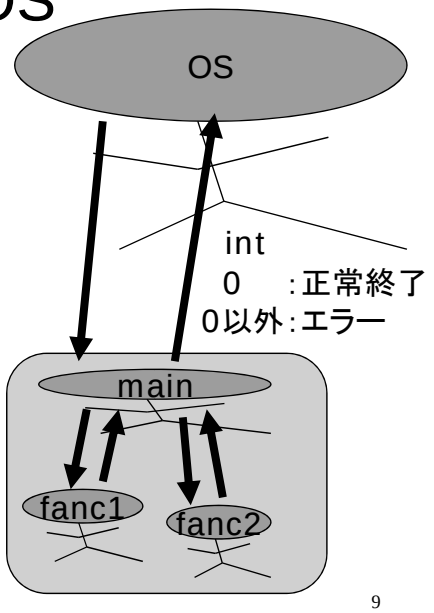
戻り値の型や関数への仮引数のリストは省略可能だが、括弧の省略はできない。

mainは特別な関数名で、一つのプログラムに必ず1つだけなければいけない。プログラムの実行はmain関数の最初から行われる。

8

関数mainの型とOS

```
/*  aaaaa.c  */
int  main()
{

      return   0;
}
```



main関数は、
OSとのやりとりを
司る大元の関数。
プログラムに必ず1つ
しかも1つだけ存在する。

9

処理の分割と関数の利用1

```
int main()
{  /*関数を用いないで組み合わせ数を計算(詳細省略) */
   for(i=1;i<n;i++){
      bunshi=bunshi*i;
   }
   for(i=1;i<m;i++){
      bunbo1=bunbo1*i;
   }
    for(i=1;i<n-m;i++){
      bunbo2=bunbo2*i;
    }
    com=bunshi/(bunbo1*bunbo2);
   return 0;
}
```

似ている。
プログラムの構造が同一。
(ひとまとまりにできない
か?)

10

処理の分割と関数の利用2

```
int main()
{
/*組み合わせ数を計算（詳細省略）*/
   bunshi=fact(n);
   bunbo1=fact(m);
   bunbo2=fact(n-m);
   com=bunshi/(bunbo1*bunbo2);
   return 0;
}
```

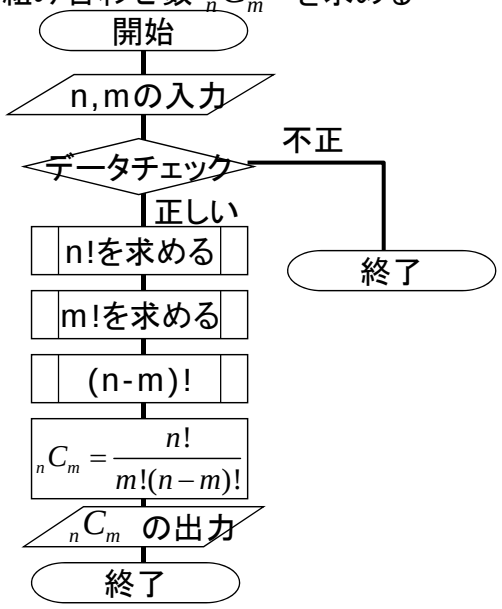
```
int fact(int k)
{
   int fac;
   for(i=1;i<k;i++){
       fac=fac*i;
   }
   return  fac;
}
```

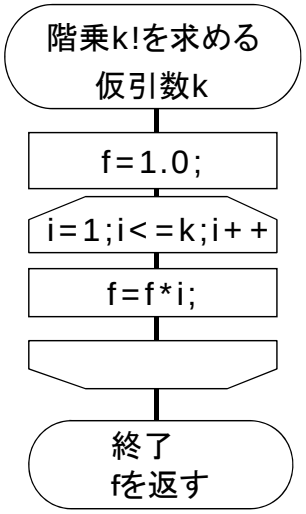
階乗を求める専門家（関数）があると、便利。

依頼データを元に、結果を返す。（階乗の計算を行なう。）

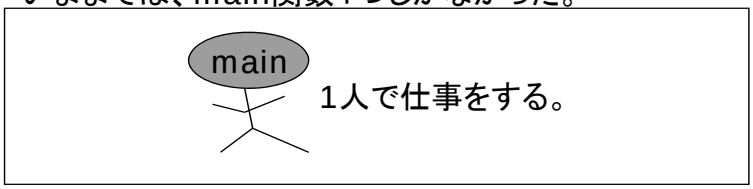
関数を表わすフローチャート

組み合わせ数 ${}_nC_m$ を求める

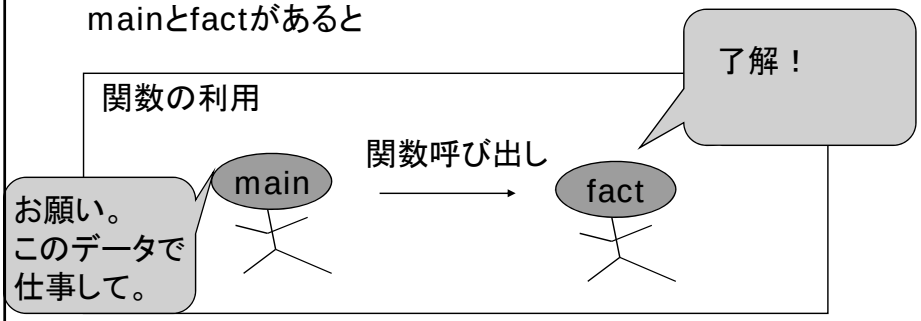




いままでは、main関数1つしかなかった。



mainとfactがあると



分業制にできる。
大きなプログラムを書くには、必要な技術。

13

戻り値の型とreturn文

書式:

```
return  式;
または、
return ;
```

関数の実行中にreturn文に出会うと、return文直後の式の値を呼び出した関数に返してその関数を終了する。

```
int   fact(int n)
{
      ****
      int   fac;
      *****
      ******
      return  fac;
}
```

14

ソースにおける関数定義位置とプロトタイプ宣言

書式

```
/*    プロトタイプ宣言       */
型1 関数名1(型a 仮引数a);          /*関数1のプロトタイプ宣言*/

/* メイン関数*/
int  main()
{
        *****
        return  0;
}

/*    関数1の定義(関数1の本体）*/
型1  関数名1(型a   仮引数a)
{
        ****
        return    型1の式;
}
```

プロトタイプ宣言と
関数定義において、
セミコロンの有無に注意
すること。

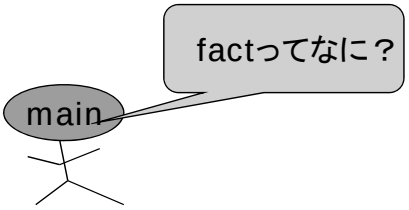
注意：main関数以外は、
プロトタイプ宣言をmain関数
前に記述する。
関数定義はmain関数後に記
述する。

15

プロトタイプ宣言の役割

プログラムは、
上から下に実行されるので、
プロトタイプ宣言が無いと。

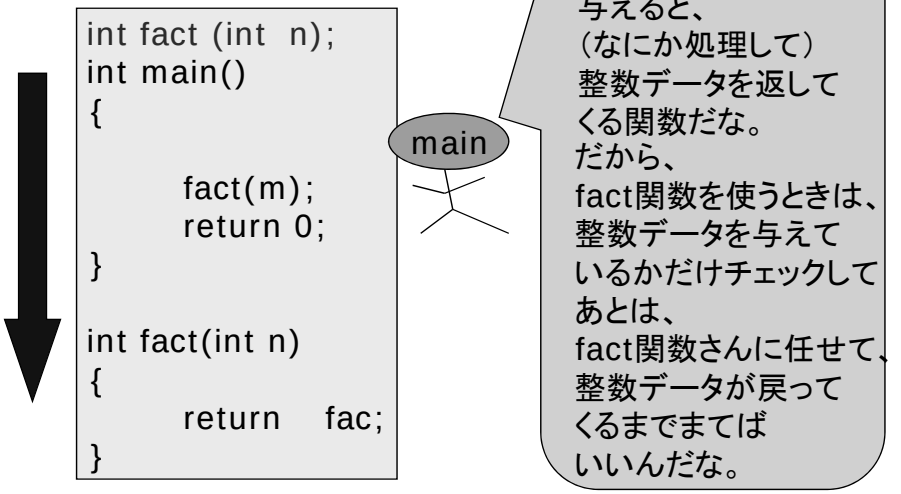
```
/**/
int main()
{

      f=fact(m);
      return   0;
}

int fact(int n)
{
      return  fac;
}
```



16

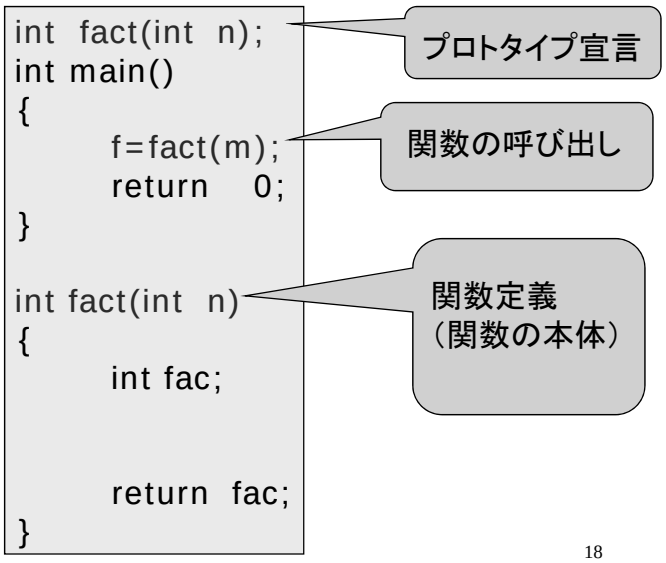
プロトタイプ宣言の役割2

プロトタイプ宣言があると。



複数の関数があるソース例

書式だけ抽出



18

実引数と仮引数

```
int  fact(int  n);
int main()
{
        f=fact(m);
        return   0;
}

int fact(int  n)
{
        int fac;


        return  fac;
}
```

呼び出す側の式(値)を
実引数(じつひきすう)と呼び、
呼び出される側の変数を
仮引数(かりひきすう)と呼ぶ。

戻り値
格納用
の変数

このmの値は実引数

この変数nは仮引数

注意:実引数が変数の場合でも、
実引数と仮引数の名前
は異なっていてもかまわない。

関数へ値の渡し方

呼び出す方では、

変数＝関数名(式); や 関数名(式);

などで関数を呼び出す。

呼び出される方では、
仮引数に実引数の値が”代入”される。

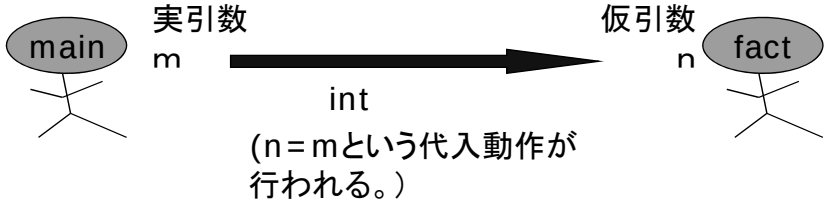
注意:実引数は変数でも
定数でも式でもよい。

```
int  main()
{
        f=fact(m);
}
int     fact(int n)
{
        int fac;
        return    fac;
}
```

mainのmの値が、
factのnに代入される。

```
int  main()
{
      f=fact(m);

      return   0;
}
```

```
int    fact(int n)
{
      return  fac;
}
```



(n＝mという代入動作が
行われる。)

21

呼び出し側への戻り値の渡し方

呼び出す方で、単に 関数名(式); とすると、
せっかくの戻り値が利用できない。

変数＝関数名(式); とすると、戻り値が変数に代入される。

```
int  main()
{
      int f;
      f=fact(m);
}

int    fact(int n)
{
      int     fac;
      return fac;
}
```

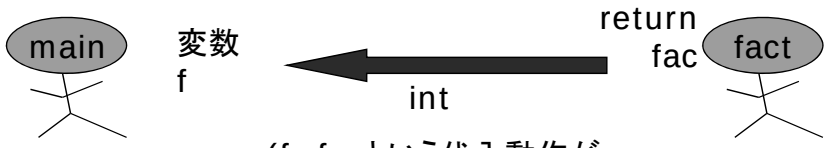
呼び出す側では、
"関数名(式)"全体を
一つの式あるいは一つの変数
のように考えてもよい。

factのfacの値が、
mainのfに代入される。

22

```
int  main()
{
      int a;
      f=fact(m);

      return  0;
}
```

```
int    fact(int n)
{
      return  fac;
}
```



(f=facという代入動作が行われる。)

23

関数呼び出しにおける制御の流れ1

制御が関数factに移ると共に、実引数(mainのm)の値が仮引数(factのn)に代入される。

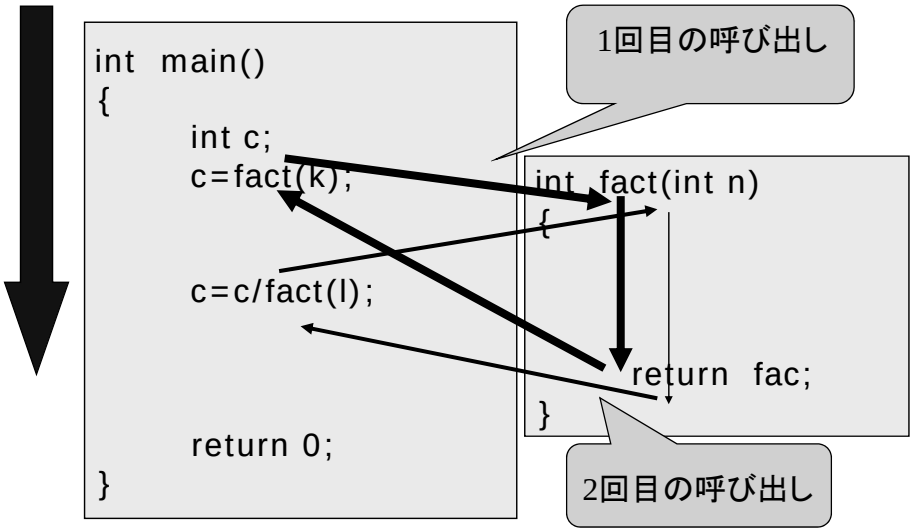
```
int  main()
{
      int f;
      f=fact(m);

      return  0;
}
```

```
int    fact(int n)
{

      return  fac;
```

制御がmain関数に移る共に、factの式facの値がmainの変数aに代入される。

24

関数呼び出しにおける制御の流れ2



同じような処理を複数回行いたいとき、
関数を用いると便利。

練習1

```
/*test_function.c　関数実験1　コメント省略*/
#include <stdio.h>
int switch_sign(int c);
int main()
{     int a;
      int b;
      printf(”整数を入力して下さい。a= ?");
      scanf("%d",&a);
      printf(“関数呼び出し前　a=%d : b= %d ¥n”,a,b);
      b=switch_sign(a);
      printf(“関数呼び出し後　a=%d : b= %d ¥n”,a,b);
      printf("%d　= switch_sign( %d) ¥n",b,a);
      return 0;
}/*　つづく*/
```

```
/*整数の符合を反転させる関数switch_sign
        仮引数c:被演算項(整数)
        戻り値:"- 被演算項"の値を返す。*/
int  switch_sign(int    c)
{

        printf("関数switch_sign実行中¥n");
        printf("c=%d¥n",c);
        return    -c;
}
```

こんな風にコメントを付けること。(スタイル規則参照。)

27

一般的な関数定義
(複数の引数を持つ複数の関数定義)

書式

```
 /*          プロトタイプ宣言       */
型1 関数名1(型1a  仮引数1a, 型1b 仮引数1b, ・・・);
型2 関数名2(型2a  仮引数2a, 型2b 仮引数2b, ・・・);
int main()
{
}

型1 関数名1(型1a  仮引数1a, 型1b  仮引数1b, ・・・)
{
}
型2  関数名2(型2a  仮引数2a, 型2b  仮引数2b,・・・)
{
}
```

プロトタイプ宣言では、セミコロンを忘れずに。

28

練習2

```
/*test_function2.c　関数実験2　コメント省略*/
/*ヘッダファイルの取り込み*/
#include <stdio.h>

/*プロトタイプ宣言*/
int  add(int x,int y);              /*  和を求める関数 */
int  sub(int x,int y);             /*  差を求める関数 */
int  main()
{
        int a;
        int b;
        int wa;
         int sa;

/*次に続く */
```

29

```
/*続き*/
        printf(”整数を入力して下さい。a= ?");
        scanf("%d",&a);
        printf(”整数を入力して下さい。b= ?");
        scanf("%d",&b);


        /*和を求める関数関数呼び出し。*/
        wa=add(a,b);
         printf("%d  = add( %d,%d) ¥n",wa,a,b);

        /*差を求める関数関数呼び出し。*/
        sa=sub(a,b);
         printf("%d  = sub( %d,%d) ¥n",sa,a,b);

        return   0;
}
```

30

```
/*続き*/
/*2つの整数の差を計算する関数add
        仮引数x:被演算項1(整数)
        仮引数y:被演算項2(整数)
        戻り値:"被演算項1+被演算項2"の値を返す。*/
int     add(int    x, int  y)
{
        /*変数宣言*/
        int   z;

        /*計算*/
        z=x+y;
        return  z;
}
/*add終了*/
/*続く*/
```

こんな風に、
関数にコメントを付けること。
(スタイル規則参照。)

31

```
/*続き*/
/*2つの整数の差を計算する関数sub
        仮引数x:被演算項1(整数)
        仮引数y:被演算項2(整数)
        戻り値:"被演算項1- 被演算項2"の値を返す。*/
int     add(int    x, int  y)
{
        /*変数宣言*/
        int   z;

        /*計算*/
        z=x+y;
        return  z;
}
/*sub終了*/
/*全プログラム(test_function2.c)終了*/
```

こんな風に、
関数にコメントを付けること。
(スタイル規則参照。)

32

組み合わせの数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:combi.c
        実行ファイル:combi
        説明:組み合わせ数nCmを求めるプログラム。
        入力:標準入力から2つの正の整数n,mを入力。
                n,mともに15以下とする。
        出力:標準出力に組み合わせ数nCmを出力。
*/
#include <stdio.h>

/* プロトタイプ宣言*/
int     fact(int);     /*階乗を計算する関数。*/
```

```
/*      続き    */
/*main関数*/
int     main()
{
        /*ローカル変数宣言*/
        int     n;      /*nCmのn*/
        int     m;      /*nCmのm*/
        int     com;            /*組み合わせ数nCm*/

/*      次のページに続く      */
```

```
    /*    続き   */
    /*    入力処理 */
    printf("組み合わせ数nCm を計算します。¥n);
    printf("Input  n=? ");
    scanf("%d",&n);
    printf("Input  m=? ");
    scanf("%d",&m);

    /* 入力値チェック */
    if(n<0||15<n||m<0||15<m||n<m)
       {
          /*不正な入力のときには、
            エラー表示してプログラム終了*/
          printf("不正な入力です。¥n");
          return  -1;
       }
    /*    正しい入力のとき、これ以降が実行される。*/
    /*    次ページへ続く       */
```

```
    /*    続き   */

    /*    組み合わせ数を計算 */
    com=fact(n)/( fact(m)*fact(n-m) );

    /*出力処理*/
    printf("%d  C %d = %5d¥n",n,m,com);

    return 0;
}
/*main関数終了*/

/*    次に続く      */
```

```
        /*      続き    */
/*階乗を求める関数
        仮引数n: n!のn(0以上15未満の値とする。)
        戻り値:n!を返す。*/
int     fact(int n)
{
        /*      ローカル変数宣言     */
        int     i;      /*ループカウンタ*/
        int     fac;    /*階乗n!*/

        fac=1;          /*0！＝1であるので1を代入*/

        /*      次に続く        */
```

37

```
        /*      続き */
        /*計算処理*/
        for(i=1;i<=n;i++)
           {
              /*階乗の計算*/
              fac=fac*i;
           }

        return fac;  /*facの値n!を戻す*/
}
/*関数factの定義終*/
/*全てのプログラム(combi.c)の終了*/
```

38

実行例

```
$make
gcc  combi.c  -o combi
$ ./combi
組み合わせ数nCm を計算します。
Input  n=? 4
Input  m=? 3
4C3  = 4
$
```

```
$./combi
組み合わせ数nCm を計算します。
Input  n=? 4
Input m=? 5
不正な入力です。
$
```