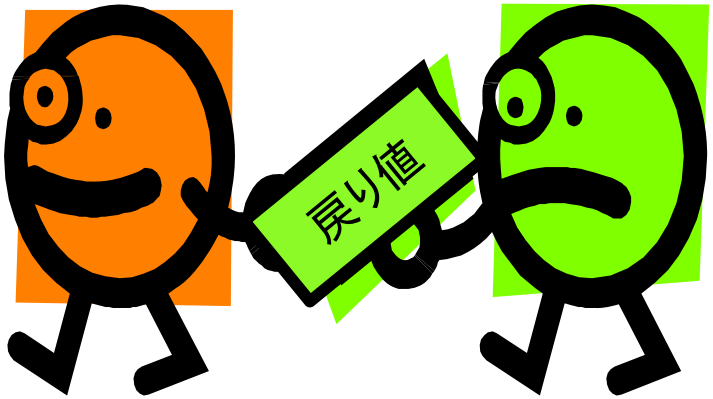
第９回関数Ⅰ
（簡単な関数の定義と利用）



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。
- 関数の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

ライブラリ関数の使い方（復習）

書式

関数名（式）

単独で使う場合

関数名（式）;

値を変数に代入する場合

変数＝関数名（式）;

ライブラリ関数：
誰かがあらかじめ作っておいてくれたプログラムの部品。
通常ヘッダファイルと一緒に用いる。
コンパイルオプションが必要なものもある。

ライブラリ関数使用例

単文として記述する関数

```
printf("辺1:¥n");
```

printf : 指定された文字列を標準出力に出力するライブラリ関数

式の中で使う関数

```
diag = sqrt(2.0)*edge*2.0;
```

sqrt:平方根を求めるライブラリ関数

今回は、ライブラリ関数 sqrt などと同じように、式の中で使うことができるような関数を自分で作る方法、使う方法について学ぶ。

標準ライブラリの数学関数（一部）

sin(x)	$\sin x$	: x の正弦
cos(x)	$\cos x$	: x の余弦
tan(x)	$\tan x$	: x の正接
log(x)	$\log_e x$	: x の（自然）対数
exp(x)	e^x	: e の x 乗（eは自然対数の底）
sqrt(x)	$\sqrt{x}$	: x の平方根
pow(x , y)	x^y	: x の y 乗
fabs(x)	$\lvert x \rvert$	: x の絶対値

x や y は
double型の式

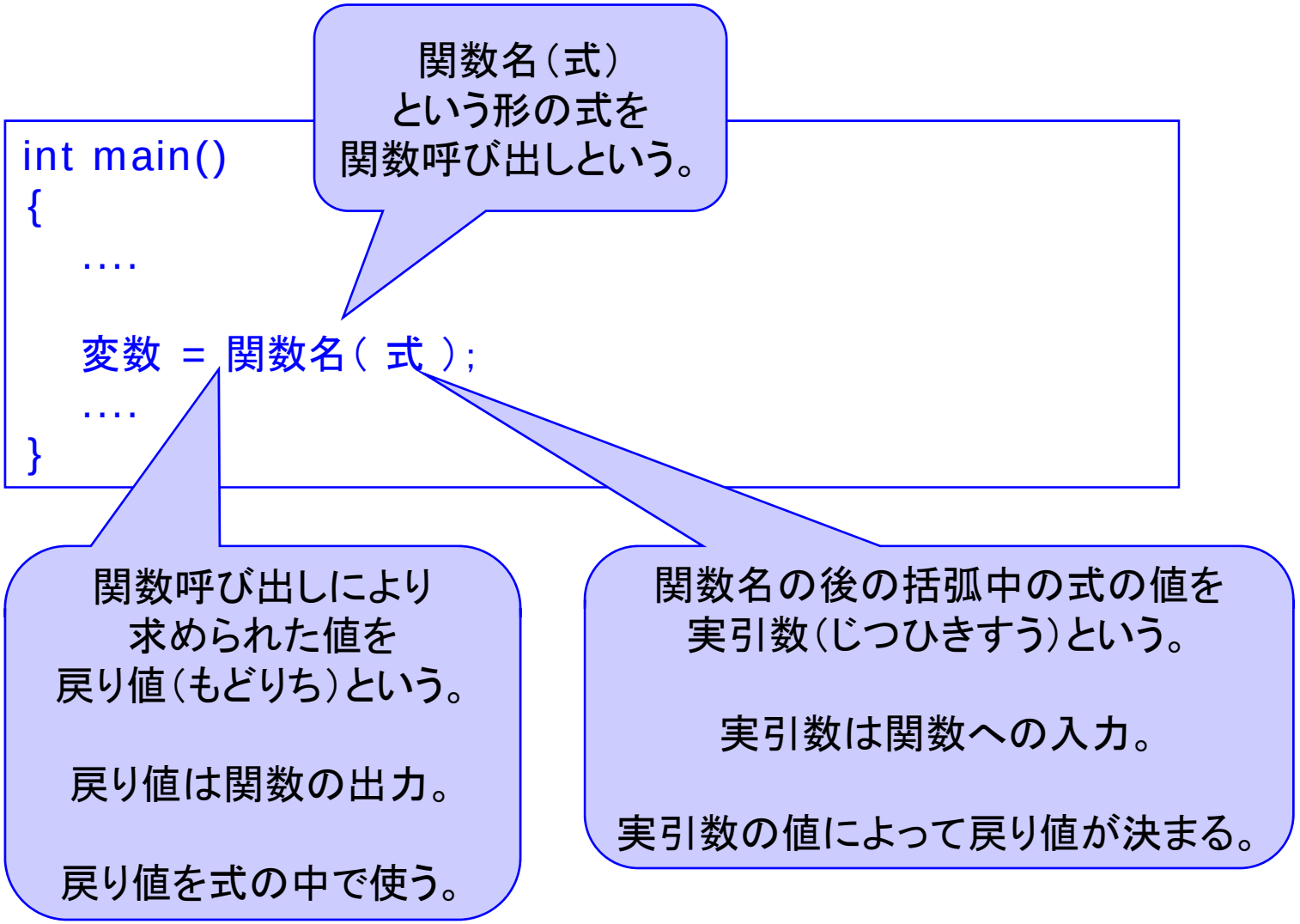
ヘッダファイル読み込み（プログラム先頭で）
#include <math.h>

コンパイル時（Makefile） mライブラリの指定
LDFLAGS = -lm

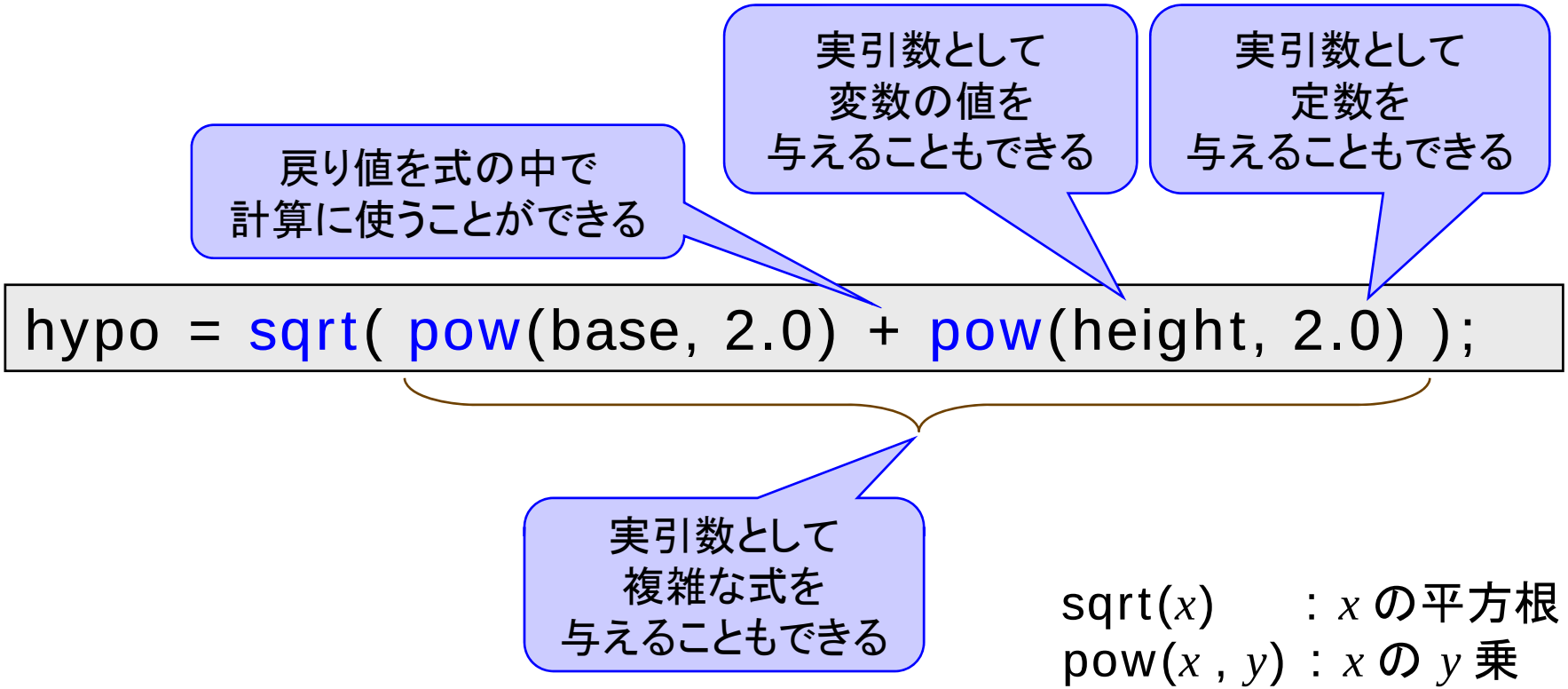
練習1

```
/*  数学関数実験  hypo1.c    コメント省略    */
/*      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 */
#include <stdio.h>
#include <math.h>

int  main()
{
   double  base;    /* 直角三角形の底辺の幅 */
   double  height;  /* 直角三角形の高さ */
   double  hypo;    /* 直角三角形の斜辺の長さ */

   printf("底辺？¥n");
   scanf("%lf",  &base);
   printf("高さ？¥n");
   scanf("%lf",  &height);

   hypo = sqrt( pow(base, 2.0) + pow(height, 2.0) );

   printf("底辺:%f , 高さ:%f のとき : 斜辺:%f ¥n",
      base, height, hypo);

   return  0;
}
```

関数の利用法（関数呼び出し）

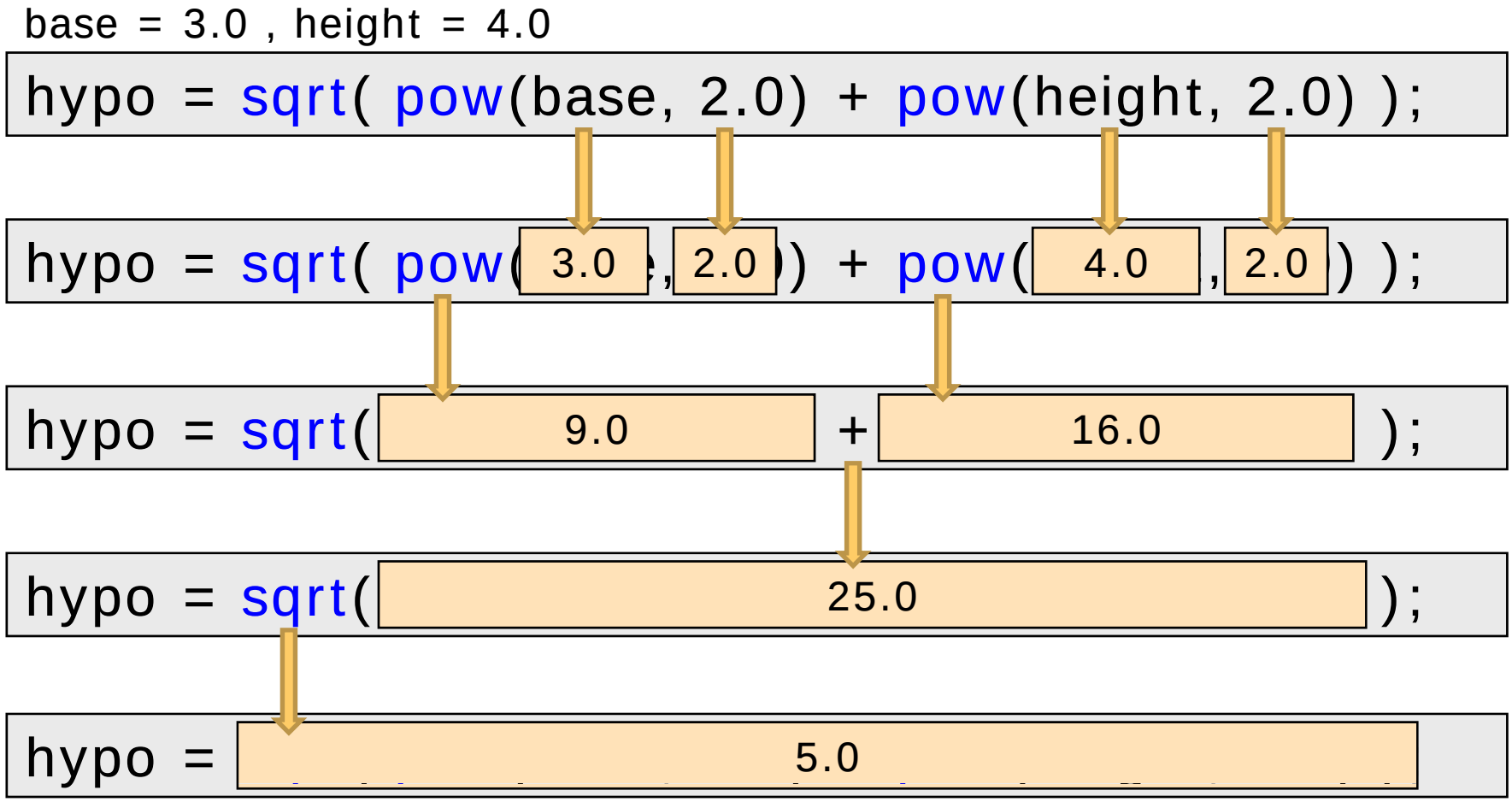


関数呼び出しを含む式



sqrt(x)　　：x の平方根
pow(x , y)：x の y 乗

関数呼び出しを含む式の計算



関数の定義

関数名は自分で命名する
（関数の意味を反映した名前にすること）

書式

```
/*  関数の説明 */
戻り値の型   関数名(仮引数の型  仮引数)
{
   return   戻り値を計算する式;
}
```

実引数の値を
受け取るための変数

仮引数の値を使って戻り値を計算する
「戻り値の型」の式

関数定義の例

```
/* 実引数を2乗した値を求める関数の定義 */
double  square(double  x)
{
   return  x*x;
}
```

自作の関数を含むプログラムの構成

書式

```
/* プログラム全体の説明 */
#include <.....>

戻り値の型　関数名(仮引数の型　仮引数); /* 関数の説明 */

int  main()
{
    *****
    return  0;
}

/*  関数の説明 */
戻り値の型　関数名(仮引数の型　仮引数)
{
    return    戻り値を計算する式;
}
```

関数のプロトタイプ宣言
（セミコロンを忘れずに！）

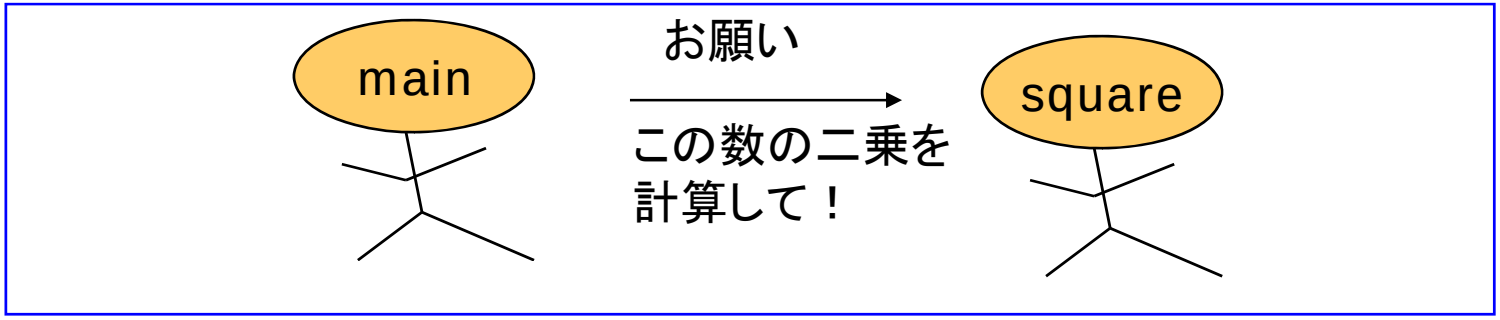
注意：
自分で作成した関数のプロトタイプ宣言を
ファイルの先頭部分（プログラム全体の説明の後）
に記述する。

関数の定義
（ここはセミコロン無し）

イメージ

以前は、main関数1つしかなかった。

main以外の関数定義があると



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。

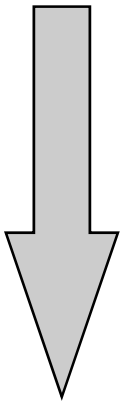
練習2

```
/*  関数定義実験  hypo2.c   コメント省略   */
/*      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 */
#include <stdio.h>
#include <math.h>


/* 関数のプロトタイプ宣言 */
double   square(double  x) ; /* 実引数を2乗した値を求める関数 */


int  main()
{
   double  base;    /* 直角三角形の底辺の幅 */
   double  height;  /* 直角三角形の高さ */
   double  hypo;    /* 直角三角形の斜辺の長さ */

   printf("底辺？¥n");
   scanf("%lf",  &base);
   printf("高さ？¥n");
   scanf("%lf",  &height);

   hypo = sqrt( square(base) + square(height) ) ;

/* 続く */
```

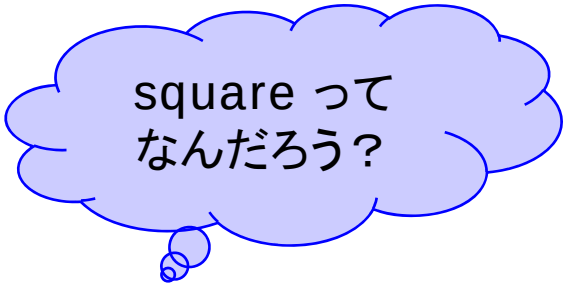
```
/*  続き */

   printf("底辺:%f , 高さ:%f のとき : 斜辺:%f ¥n",
      base,  height,  hypo);

   return  0;
}


/* 実引数を2乗した値を求める関数の定義 */
double   square(double  x)
{
   return  x*x ;
}
```

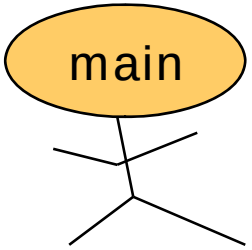
プロトタイプ宣言の役割

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。



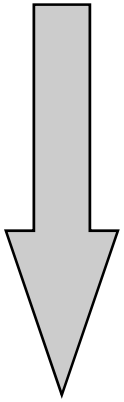
```
int main()
{

   ･･･
   hypo = ･･･ square(base) ･･･ ;
   ･･･
   return   0;
}

double   square(double x)
{
    return  x*x;
}
```



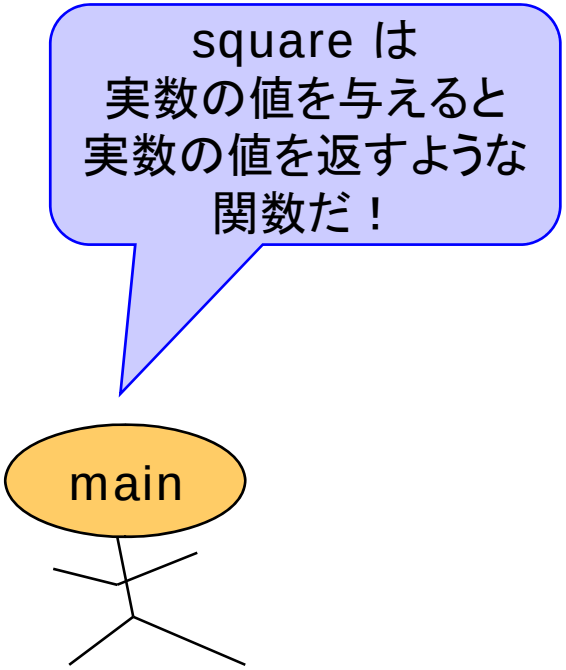


プロトタイプ宣言の役割

プロトタイプ宣言があると、
関数であることがわかる。



```
double  square(double x);

int main()
{

   ...
   hypo = ... square(base) ... ;
   ...
   return   0;
}

double  square(double x)
{
   return x*x ;
}
```



自作関数の呼び出しを含む式

```
double square(double x)
{
   return   x*x;
}
```

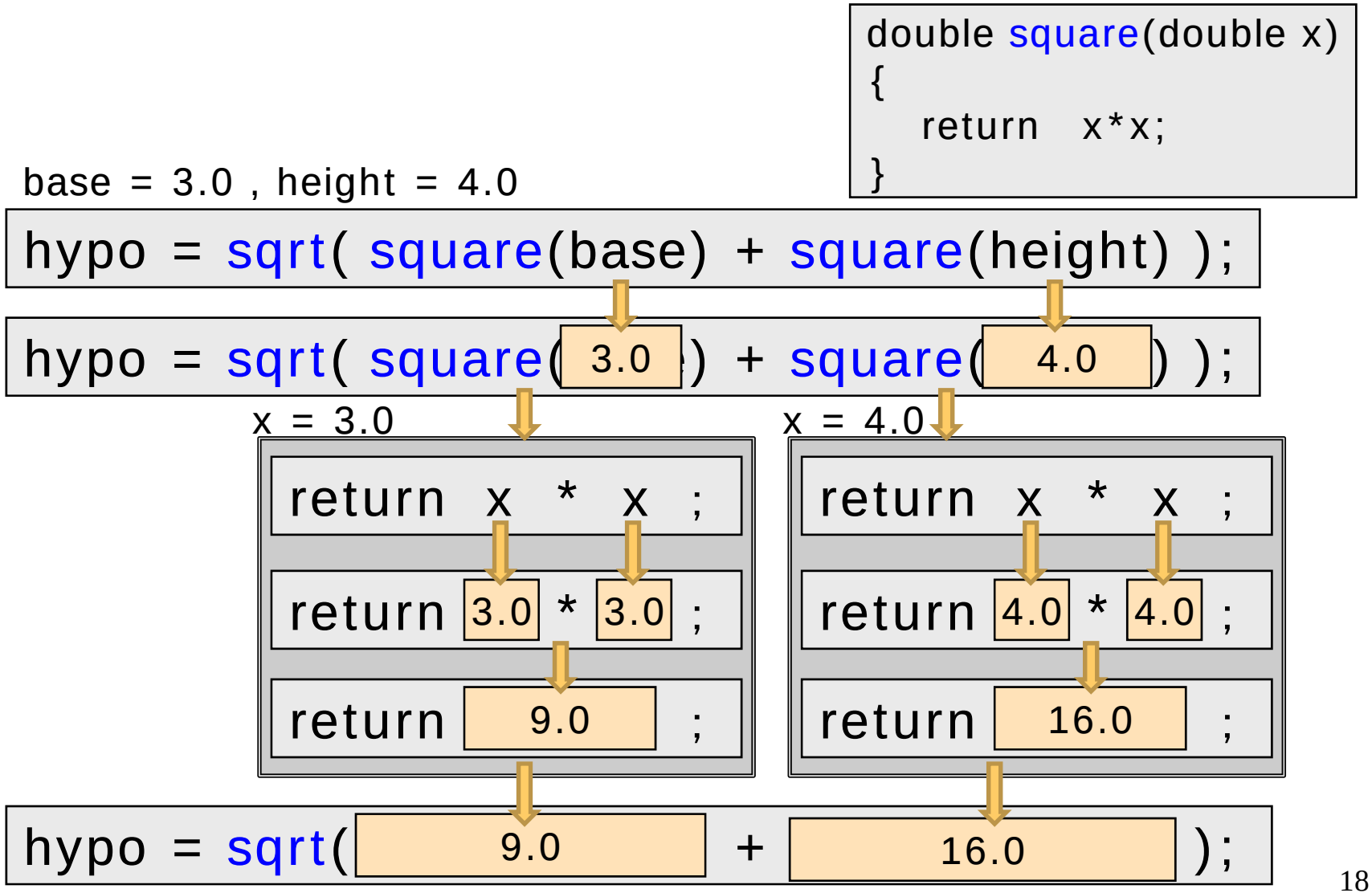
仮引数の名前は
呼び出し時は気にしない

関数呼び出し時に指定された実引数の値が
仮引数に代入されて、戻り値の計算が行われる。

```
hypo = sqrt( square(base) + square(height) );
```

実引数には変数、定数など
任意の式を与えることができる

関数呼び出しを含む式の計算



複数の引数を持つ関数の定義

書式

```
/*  関数の説明 */
戻り値の型　関数名(仮引数1の型　仮引数1, 仮引数2の型　仮引数2, ... )
{
        return　戻り値を計算する式;
}
```

仮引数の値を使って戻り値を計算する
「戻り値の型」の式

関数定義の例

```
/* 二つの実数値の2乗和を求める関数の定義*/
double  sqsum(double a, double b)
{
   return  square(a)+square(b) ;
}
```

練習3

```
/*  関数定義実験  hypo3.c    コメント省略    */
/*      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 */
#include <stdio.h>
#include <math.h>

/* 関数のプロトタイプ宣言 */
double   square(double  x) ; /* 実引数を2乗した値を求める関数 */
double   sqsum(double  a, double b) ;
/*二つの実数値の2乗和を求める関数*/

int  main()
{
   double  base;    /* 直角三角形の底辺の幅 */
   double  height;  /* 直角三角形の高さ */
   double  hypo;    /* 直角三角形の斜辺の長さ */

   printf("底辺？¥n");
   scanf("%lf",  &base);
   printf("高さ？¥n");
   scanf("%lf",  &height);

   hypo = sqrt( sumsq(base, height) ) ;

/* 続く */
```

```
/*  続き */

   printf("底辺:%f , 高さ:%f のとき : 斜辺:%f ¥n",
      base,  height,  hypo);

   return  0;
}


/* 実引数を2乗した値を求める関数の定義 */
double   square(double  x)
{
   return  x*x ;
}

/* 二つの実数値の2乗和を求める関数の定義*/
double  sqsum(double a, double b)
{
   return   square(a)+square(b) ;
}
```

平面上の線分の長さ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: lineseg2d.c
    実行ファイル: lineseg2d
    説明:平面上の線分の長さを求めるプログラム。
        数学関数を利用するため、コンパイルオプション –lm が必要。
    入力:標準入力から二次元平面上の2つの点(p、qとする)の座標を入力。
        点pのx座標、y座標、点qのx座標、y座標の順に
        4個の実数値(doubleで扱える任意の値)を空白または改行で区切って入力する。
        ただし、点pと点qは等しい座標を持つ点ではないものとする。
    出力:標準出力に点pと点qを二つの端点とする線分の長さ(正の実数値)を出力。
*/
#include <stdio.h>
#include <math.h>

/* 関数のプロトタイプ宣言 */
double  square(double  x) ; /* 実引数を2乗した値を求める関数 */
double  sqsum(double  a, double b) ; /*二つの実数値の2乗和を求める関数*/
double  distance(double x1, double y1, double x2, double y2) ;
/* 点 (x1,y1) と点 (x2,y2) の間の距離を求める関数 */

/*  次に続く */
```

```
/* 続き */

int  main()
{
   double  p_x;    /* 一方の端点(点p)のx座標  */
   double  p_y;    /* 一方の端点(点p)のy座標  */
   double  q_x;    /* 他方の端点(点q)のx座標  */
   double  q_y;    /* 他方の端点(点q)のy座標  */
   double  length_pq; /* 線分pqの長さ */

   printf("点pの座標?¥n");
   scanf("%lf",  &p_x);
   scanf("%lf",  &p_y);
   printf("点qの座標?¥n");
   scanf("%lf",  &q_x);
   scanf("%lf",  &q_y);

   /* 入力値チェック */
   if ( p_x == q_x  && p_y == q_y )
     {
       /*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("与えられた二点の座標が等しいため、");
       printf("この二点を端点とする線分は存在しません。¥n");
       return  -1;
     }
   /* 正しい入力のとき、これ以降が実行される。*/

/*  次に続く */
```

```
/*  続き  */

   length_pq = distance(p_x, p_y, q_x, q_y); /* 線分の長さは端点間の距離に等しい */

   printf("点p : (%6.2f,%6.2f) ¥n",  p_x, p_y);
   printf("点q : (%6.2f,%6.2f) ¥n",  q_x, q_y);
   printf("線分pqの長さは%6.2f です。¥n",  length_pq);

   return  0;
}
/* main関数終了 */

/* 実引数を2乗した値を求める関数
   仮引数 x : 2乗すべき値（任意の実数値）
   戻り値     : xの2乗(非負の実数値)を返す。
*/
double   square(double  x)
{
   return  x*x ;
}
/* 関数 square の定義終 */

/*  次に続く */
```

```
/*  続き  */

/* 二つの実数値の2乗和を求める関数仮引数
   仮引数  a : 2乗和を求めるべき値（任意の実数値）
   仮引数  b : 2乗和を求めるべき値（任意の実数値）
   戻り値    : aの2乗とbの二乗の和(非負の実数値)を返す。
*/
double  sqsum(double a, double b)
{
   return  square(a)+square(b) ;
}
/* 関数 sqsum の定義終 */

/* 点 (x1,y1) と点 (x2,y2) の間の距離を求める関数
   仮引数  x1 : 一方の点のx座標(任意の実数値)
   仮引数  y1 : 一方の点のy座標(任意の実数値)
   仮引数  x2 : 他方の点のx座標(任意の実数値)
   仮引数  y2 : 他方の点のy座標(任意の実数値)
   戻り値    :点 (x1,y1) と点 (x2,y2) の間の距離(非負の実数値)を返す。
 */
double  distance(double x1, double y1, double x2, double y2)
{
   return  sqrt(sqsum(x2-x1, y2-y1) ) ;
}
/* 関数 distance の定義終 */

/* 全てのプログラム(lineseg2d.c)の終了 */
```

実行例

```
$make
gcc  lineseg2d.c  -o lineseg2d

$ ./lineseg2d
点pの座標?
-2   1
点qの座標?
1   5
点p：( -2.00,  1.00)
点q：(  1.00,  5.00)
線分pqの長さは  5.00 です。

$
```