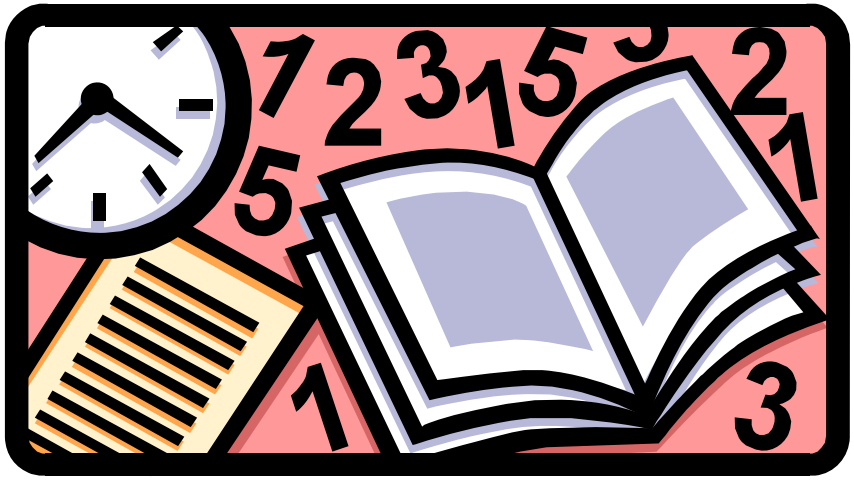
第4回簡単な計算・プリプロセッサ

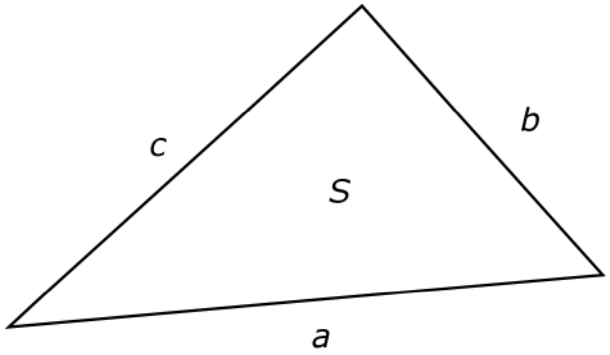


今回の目標

- 定数を理解する。
- 演算子とその効果を理解する。
- 簡単なライブラリ関数の使用法を理解する。

☆ヘロンの公式を用いた3角形の面積を求めるプログラムを作成する。

ヘロンの公式



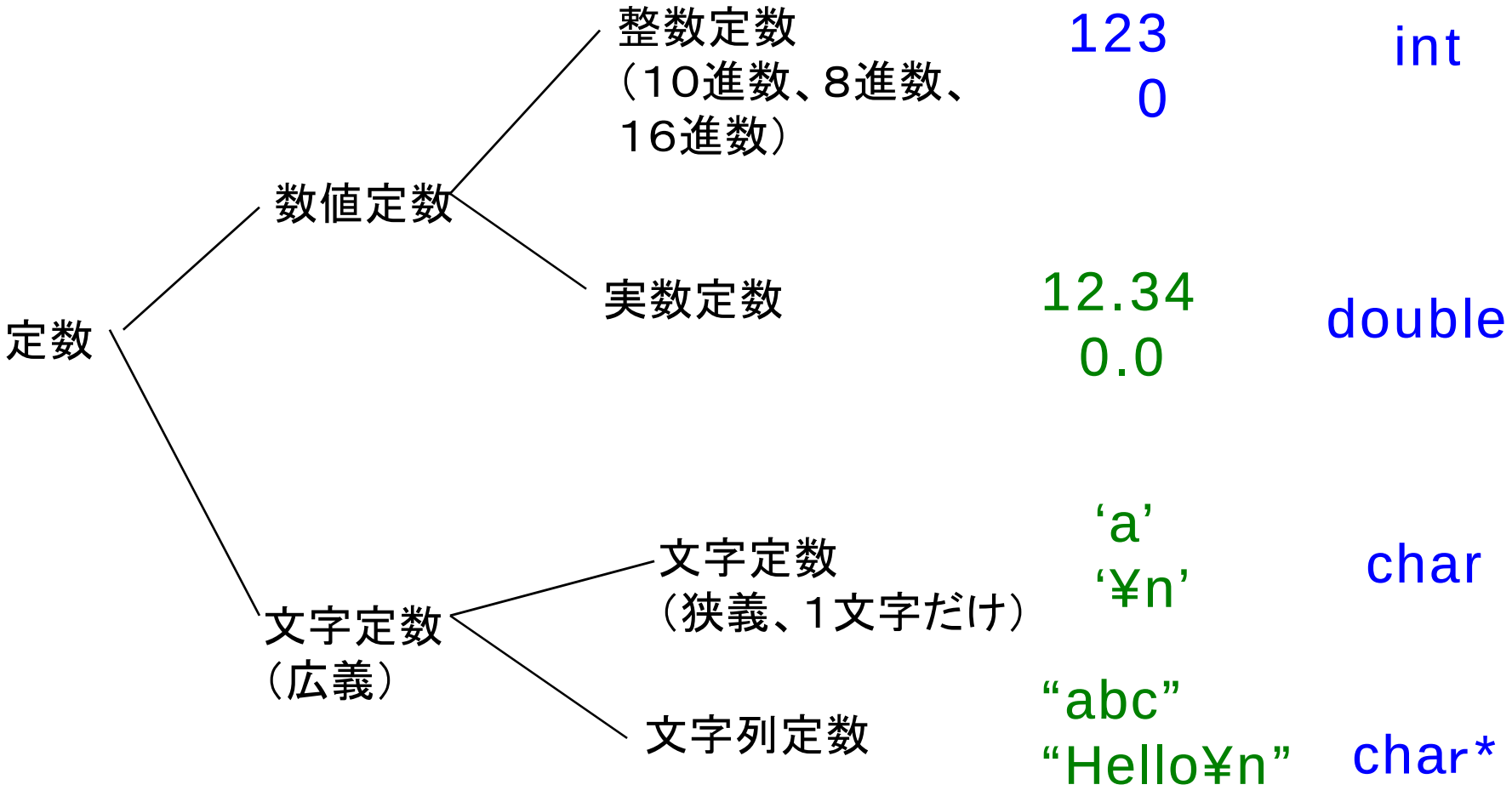
ヘロンの公式:
3辺の長さがわかっているときに、
3角形の面積を求める方法。

$$d = \frac{a + b + c}{2}$$

$$S = \sqrt{d\,(d - a)(d - b)(d - c)}$$

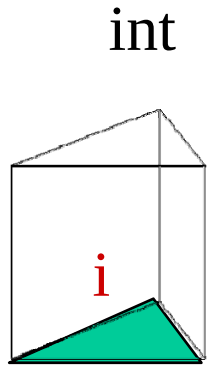
定数の分類と型

定数：プログラム中で、常に特定の値をもつ。



プログラム（ソース）内での整数定数の表現

0以外で始まる数字だけの列	10進数	123
0で始まる数字だけの列	8進数	0173
	（10進数	123）
0xで始まり、残りが数字だけの列	16進数	0x7B
	（10進数	123）



次の3つは、コンピュータ内ではまったく同じ処理を行う。

```
int     i;
i=123;
```

```
int     i;
i=0173;
```

```
int     i;
i=0x7B;
```

練習1

```
/*     整数定数実験  intconst.c  コメント省略  */
#include <stdio.h>
int  main()
{
        int     i;

        i=123;
        printf("%d¥n",i);
        i=0173;
        printf("%d¥n",i);
        i=0x7B;
        printf("%d¥n",i);

        return 0;
}
```

プログラム（ソース）内での実数定数の表現

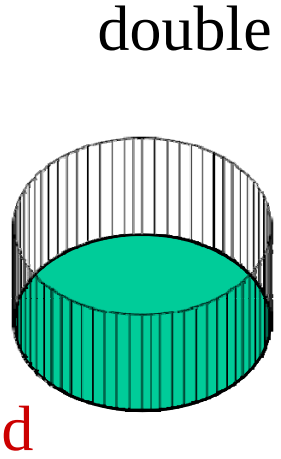
小数点を含む数字列　12.34

e、Eを含む形　1234e-2

（意味　$1234\times10^{-2}=12.34$　）

e、Eと小数点を含む形　0.1234E+2

（意味　$0.1234\times10^{+2}=12.34$　）



次の3つは、コンピュータ内ではまったく同じ処理を行う。

```
double d;
d=12.34;
```

```
double d;
d=1234e-2;
```

```
double d;
d=0.1234E+2;
```

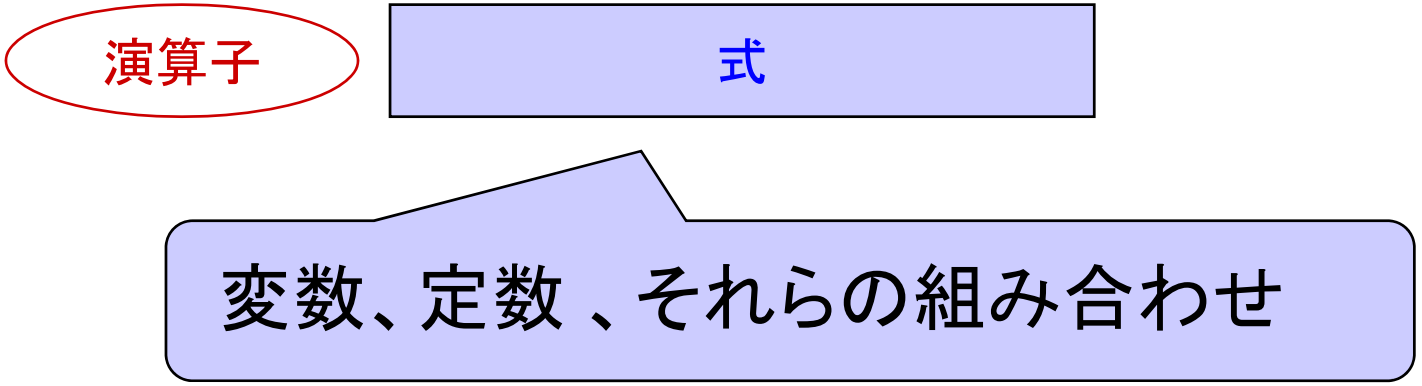
練習2

```
/*実数定数実験　realconst.c　コメント省略　　*/
#include <stdio.h>
int  main()
{
　　double　　d;

　　d=12.34;
　　printf("%6.2f¥n",d);
　　d=1234e-2;
　　printf("%6.2f¥n",d);
　　d=0.1234E+2;
　　printf("%6.2f¥n",d);

　　return 0;
}
```

演算子

C言語では、演算子と式（変数、定数等）を組み合わせて、プログラムが記述される。

単項演算子の書き方（前置型）



単項演算子の書き方（後置型）

二項演算子の書き方

算術演算子（C言語での算術計算）

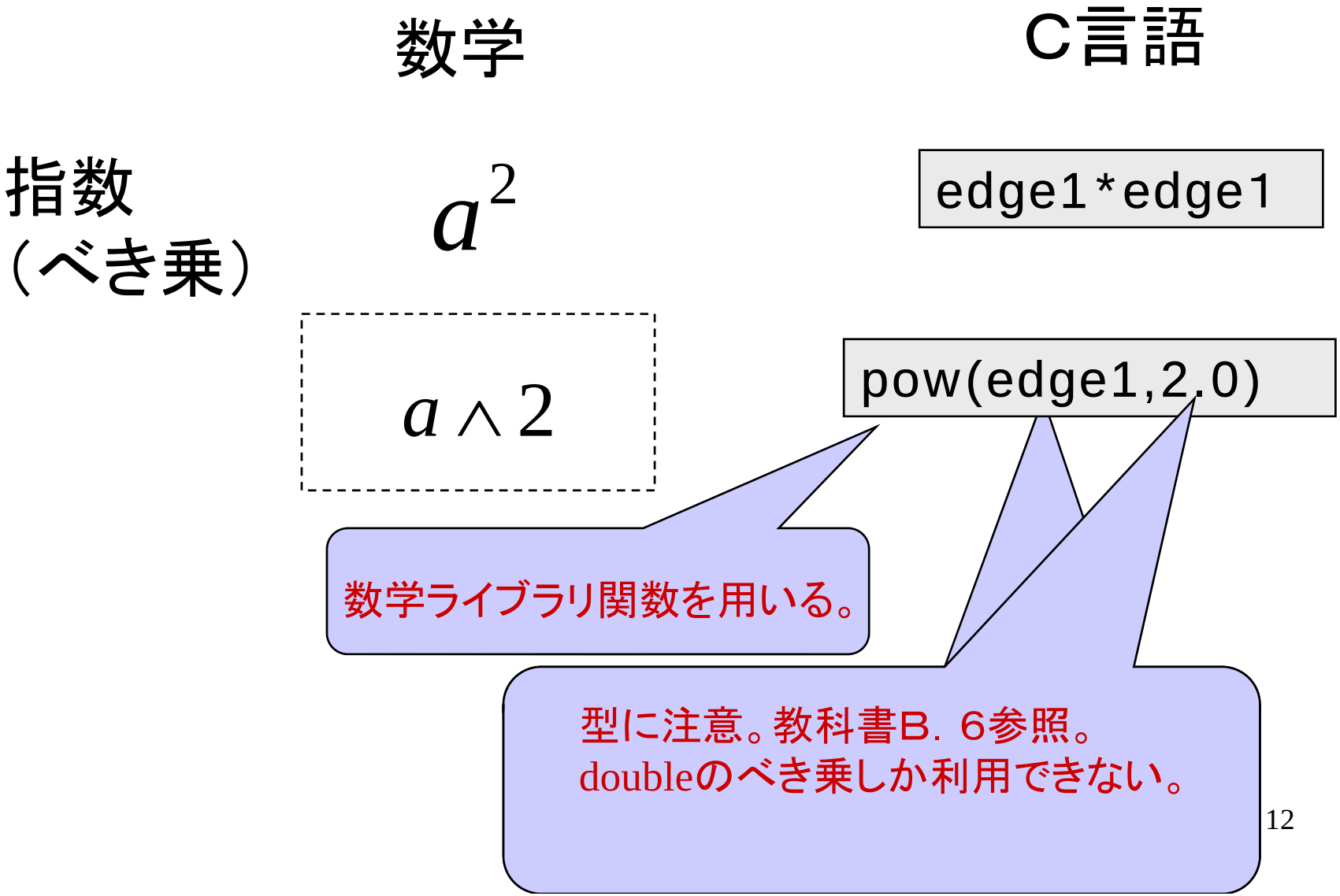
	記号	例	意味
単項演算子	-	-a	aの値と-1の積
2項演算子 （四則演算とモジュロ演算）	+	a+b	aの値とbの値の和
	-	a-b	〃　差
	*	a*b	〃　積
	/	a/b	〃　商
	%	a%b	aの値をbの値で割ったときの余り

数学との表記の違い1

	数学	C言語
掛け算	$a \times b$	`edge1*edge2`
	$a \bullet b$	
	ab（記号省略可）	`edge1edge2`

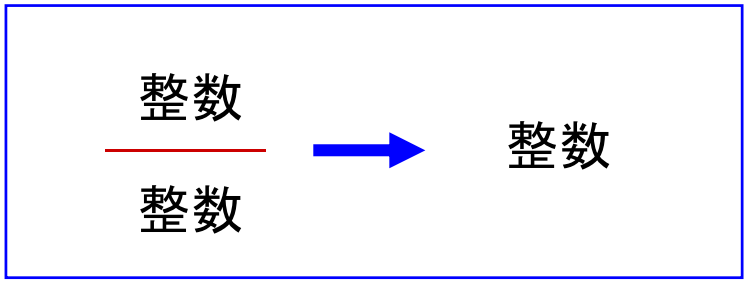
間違い
（演算子省略不可）
この記述だと、長い変数名だと思われる。

数学との表記の違い2


数学
C言語
指数
（べき乗）
a^2
edge1*edge1
$a \wedge 2$
pow(edge1,2.0)
数学ライブラリ関数を用いる。
型に注意。教科書B．6参照。
doubleのべき乗しか利用できない。

結合規則と細則

[規則]整数どうしの割り算では、商の小数部分は切り捨てられる。




```
int  i;
int j;
double  x;
i  = 7;
j  = 2;
x =  i / j;
```

上のようなプログラムでは、
xの値は3．0である。

```
double      taiseki;
double      takasa;
double  teimen;

taiseki=1/3*takasa*teimen;
```

上のようなプログラムでは、
takasaとteimenの値に関わらず、
taisekiの値は0.0である。

結合規則と細則

[規則]演算子%は、double型には適用できない。

```
a=b%c;
```

余りを求める演算。

このような例では、a,b,cすべてが整数（int型）でなくてはならない。

例

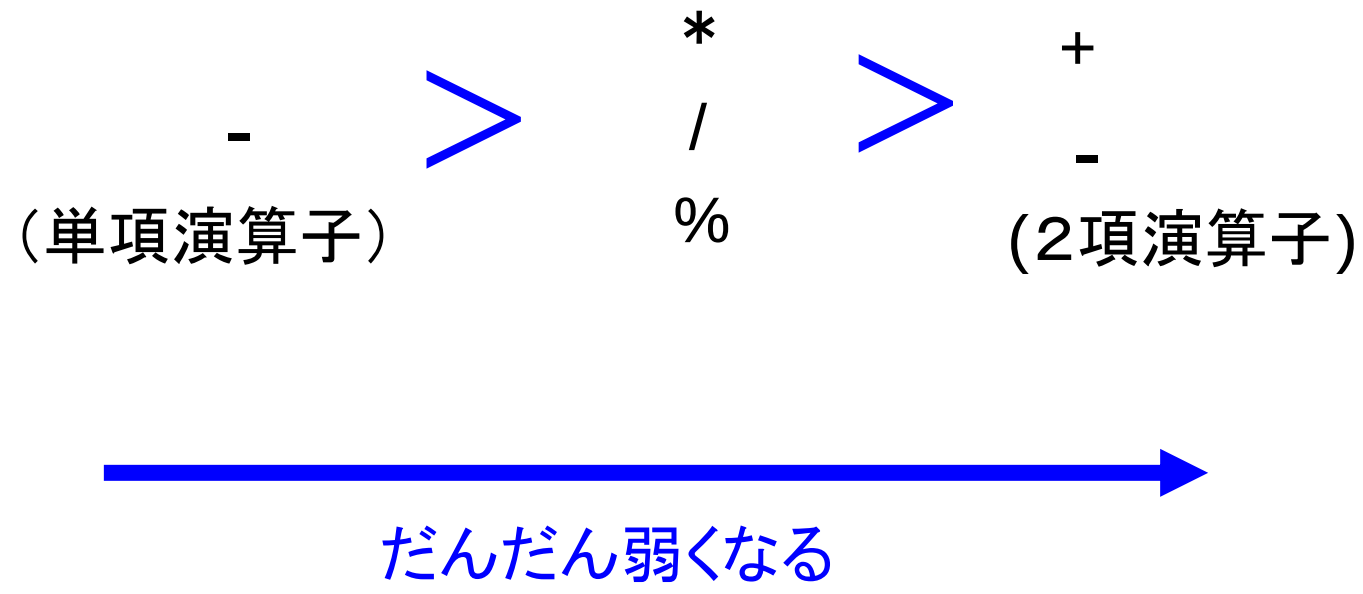
```
x=7/2;
y=7%2;
```

7÷2は、商が 3 で余りが 1 である。

x　　y

結合規則と細則

[規則]演算子の結合力は以下のとおり。同じ結合力のときは、左から計算される。また、括弧で計算順序を指定できる。



結合力の例

意味

```
x =  a  / b *c;
```

→ x = (a/b) *c ○

x = a/(b *c)とは違う。 ×

```
y =  -a - b*-c;
```

→ y = (-a) -(b*(-c)) ○

結合力が不安であれば、
括弧をつかって明示した方がいい。

```
x = (a/b) *c;
```

```
y = (-a) -(b*(-c));
```

実験3

```
/*      結合力実験  priority.c    コメント省略     */
#include <stdio.h>
int  main()
{       int     a;
        int     b;
        int     c;
        int     d;
        int     x;
        int     y;
        int     z;
        a=5;
        b=4;
        c=3;
        d=2;
        x=0;
        y=0;
        z=0;
        /*つづく*/
```

```
        /*つづき*/
        x=-a+b/c*d;

        y=-(a+b/c*d);

        z=-((a+b)/c*d);


        printf("x= %3d ,   y=%3d,  z=%3d ¥n",x,y,z);

        return 0;
}
```

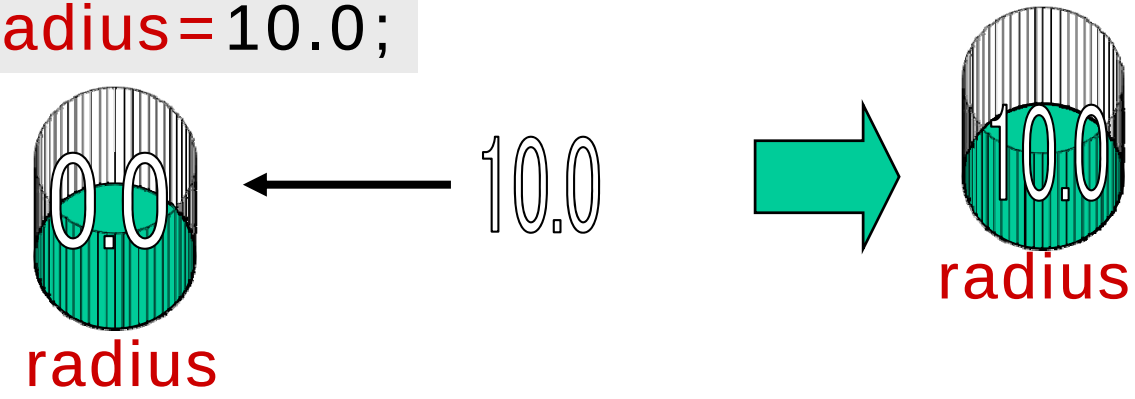
代入演算子

C言語では「＝」は代入演算子(2項演算子)である。
(数学の「＝」とは違う意味)

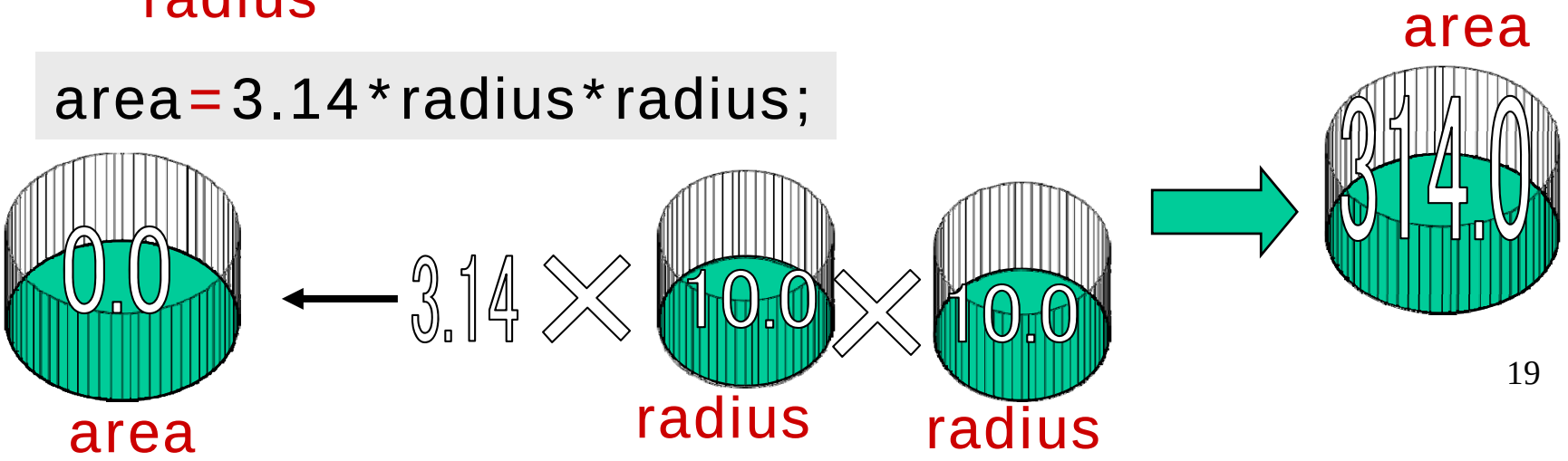
変数＝式

の形でつかわれる。
左辺は必ず変数で、右辺は式(定数や算術式等)。

```
radius=10.0;
```




```
area=3.14*radius*radius;
```



型の表現能力

型の表現能力

高い＝低い

double = int;

→ 問題なし

低い＝　高い

int = double;

→ 切り捨てが起こる。

本演習では、
代入演算子（＝）において左辺の型と右辺の型は必ず
同じにすること。（スタイル規則E-3　参照）

どうしても、違う型になるときは、
キャスト演算子を用いること。

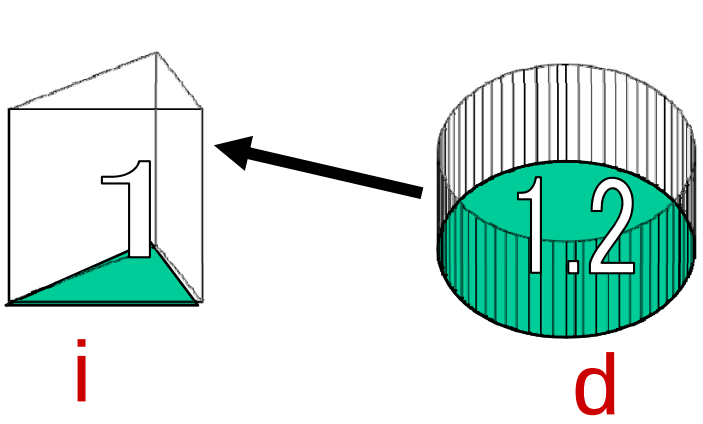
キャスト演算子（型の変換法）

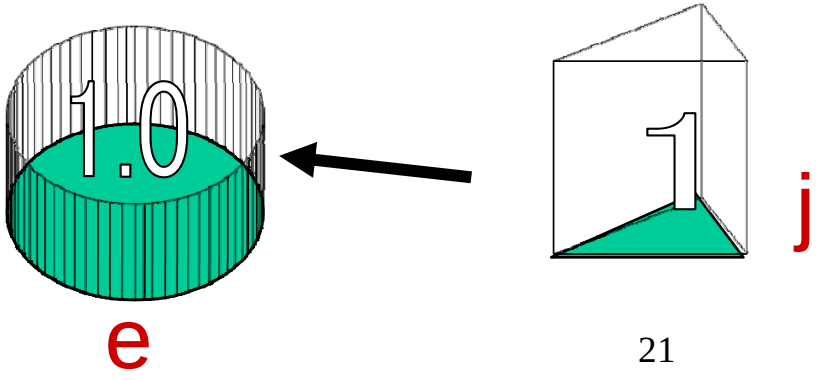
キャスト演算子は型を変換する。
（スタイル規則参照）

書式 （データ型）式

```
int          i;
double    d;
d=1.2;
i = (int) d;
```

```
int   j;
double    e;
j=1;
e = (double) j;
```





異なる型同士の演算

[規則]2項演算子の被演算項の型が異なる場合には、
表現能力の低い方を高い方に変換してから演算が行われ、
演算結果は表現能力の高い方になる。

```
int    a;
double b;
double c;
c=a*b;
```

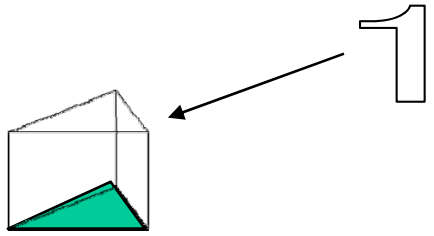
```
int    a;
double b;
double c;
c= ((double) a) *b;
```

本演習では、
このような自動型変換は用いない事。
（スタイル規則参照）

インクリメント演算子とデクリメント演算子

C言語では、1つづつの増減用に、
簡単な形が用意されている。

インクリメント演算子 ++

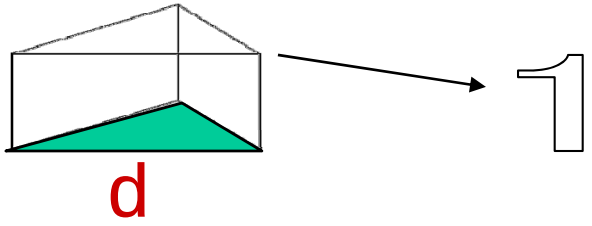


c++;

別の書き方。

c=c+1;

デクリメント演算子 --

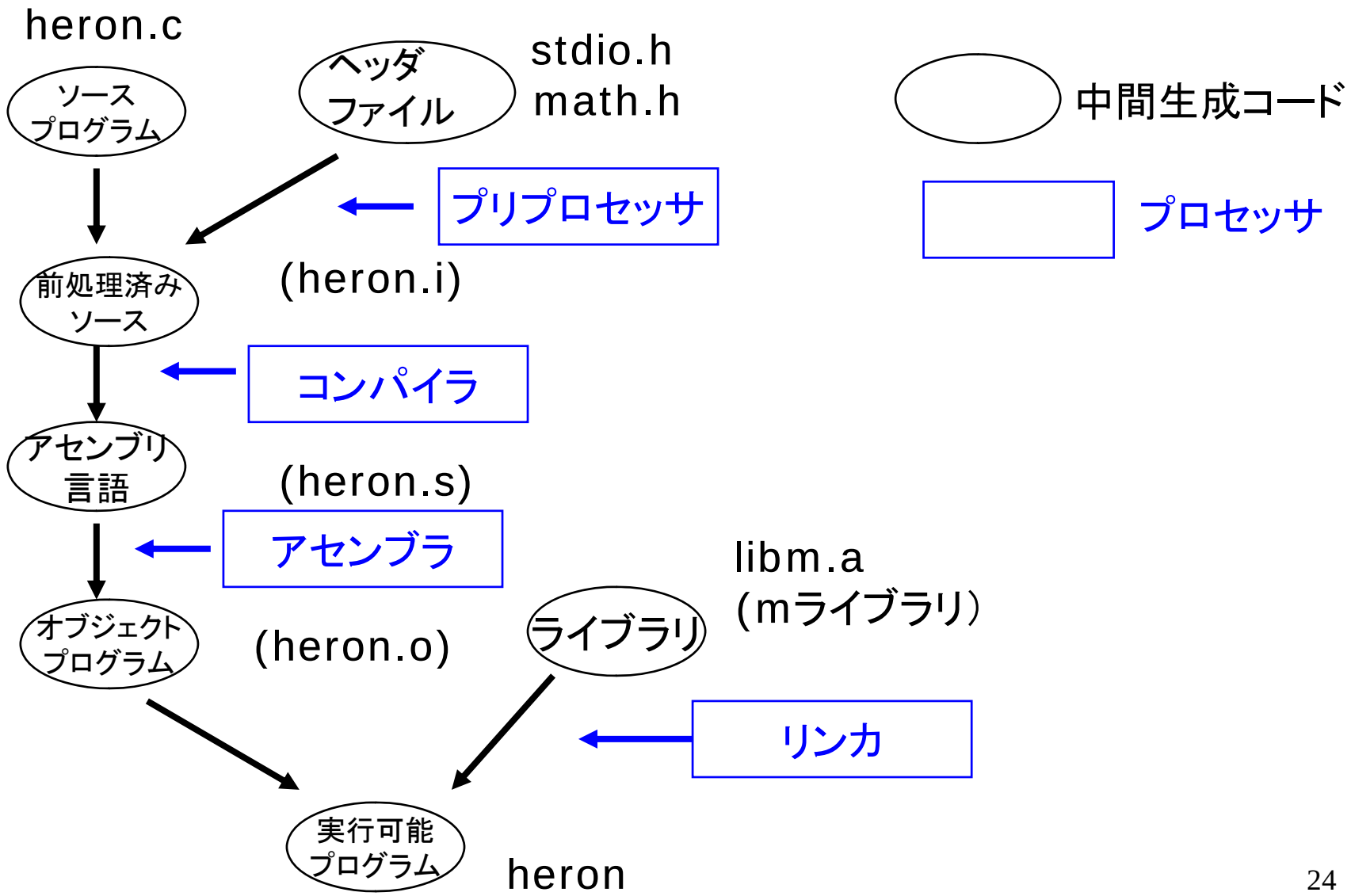


d--;

別の書き方。

d=d-1;

gccコマンドの詳細


heron.c
ソース
プログラム
ヘッダ
ファイル
stdio.h
math.h
中間生成コード
プリプロセッサ
プロセッサ
前処理済み
ソース
(heron.i)
コンパイラ
アセンブリ
言語
(heron.s)
アセンブラ
libm.a
(mライブラリ)
オブジェクト
プログラム
(heron.o)
ライブラリ
リンカ
実行可能
プログラム
heron

プリプロセッサへの指示

[規則]プリプロセッサへの指示行は、
必ず # ではじまる。

例

```
#include <stdio.h>
#include <math.h>

#define MAX 1000
```

注意：プリプロセッサ行は
行末にセミコロン「;」をつけない。

#define

```
#define　文字列1 文字列2
```

ソースのなかの文字列1を文字列2に書き換える（置換する。）

この定義をマクロ定義と呼び、
文字列1をマクロ名、
文字列2をマクロ展開という。

通常の変数と区別するため、本演習ではマクロ名に英小文字を用いないこと。（スタイル規則参照）

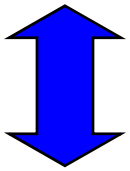
例

```
#define  M_PI   3.14159265358979323846   /*  pi  */
```

math.hの中でこう定義されている。

#difine例

```
#define    EPS  (1.0e-5)
int main()
{
        .
        .
        yukou=data+EPS;
```



同じ効果

```
int main()
{
        .
        .
        yukou=data+(1.0e-5);
```

意味の分かりずらい
数値だけの記述
（マジックナンバー）は、
できるだけ避けること。

#include

他のファイルをソースファイル内に読み込む。
読み込まれるファイルをヘッダファイルという。
ヘッダファイルの拡張子は、
.h
である。

書式

#include　<ファイル名>　→　システムが用意しているヘッダファイルを取り込む

#include　"ファイル名"　→　自分で作ったヘッダファイルなどを取り込む

代表的なヘッダファイル

stdio.h:　標準入出力用
（ printf() ,scanf()等が使えるようになる。）

string.h:文字列処理用
（strcpy(),strcmp()等が使えるようになる。）

stdlib.h:数値変換、記憶割り当て用
（atof(),malloc()等が使えるようになる。）

math.h:数学関数用
(sin(),cos(),sqrt()等の数学ライブラリ関数が
使えるようになる。
mライブラリと一緒に使う。）

（ヘッダファイルで宣言している関数を用いるには、
コンパイルオプションが必要なものもある。
コンパイルの仕方の概略をコメントしておくとよい。)

ライブラリ関数の使い方

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）；

詳しくは、第９回で説明する。

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

単独で記述する場合

```
printf("辺1:¥n");
```

()内の文字列を標準出力に出力するライブラリ関数

他の演算子と組み合わせる場合

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

同じ効果

sqrt:平方根を求めるライブラリ関数

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

Makefile の記述追加

いままでのMakefile

```
CC　= gcc
all: 実行ファイル名（ソースファイルから.cを除いたもの）
```

これだと、コンパイルオプションがないので、数学関数ライブラリ（mライブラリ）を用いるソースをコンパイルできない。

数学ライブラリを用いるときのMakefileは以下のように記述する。

```
CC　= gcc
LDFLAGS＝－lm
all: 実行ファイル名（ソースファイルから.cを除いたもの）
```

（ガイダンス資料参照）

Makefile 例

CC　= gcc
LDFLAGS＝－lm
all: 実行ファイル名（ソースファイルから.cを除いたもの）

Makefile

```
CC=gcc
LDFLAGS=-lm
all:heron
```

3角形の面積を求めるプログラム(p.56参照)

```
/*
        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:heron.c
        実行ファイル:heron
        説明:ヘロンの公式を用いて3角形の面積を求めるプログラム
              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3辺の長さを入力する。
              各入力は、正の実数とし、
              どの順序に入力されてもよい。
        出力:与えられた3辺の長さを持つ三角形の面積を
              標準出力に出力する。面積は正の実数。
*/
/*      プログラム本体は次のページ以降   */
```

```
#include <stdio.h>
#include <math.h>
int  main()
{
        /*      変数宣言                */
        double      edge1;      /*辺1の長さ*/
        double      edge2;      /*辺2の長さ*/
        double      edge3;      /*辺3の長さ*/

        double      heron_d;    /* ヘロンの公式用
                                    の一時保存用の変数*/
        double      area;       /*3角形の面積*/

/*      次ページへ続く          */
```

```
        /*      入力処理        */
        printf("辺1の長さ:¥n");
        scanf("%lf",&edge1);
        printf("辺2の長さ:¥n");
        scanf("%lf",&edge2);
        printf("辺3の長さ:¥n");
        scanf("%lf",&edge3);

/*      次ページへ続く          */
```

```
        /*      計算処理      */
        heron_d=(edge1+edge2+edge3)/2.0;

        area=sqrt(heron_d*(heron_d-edge1)*
                (heron_d-edge2)*(heron_d-edge3));

        /*      出力処理      */
        printf("3角形の面積は　%6.2f　です。¥n",area);

        return 0;
}
```

コンパイルと実行

```
$gcc -lm heron.c  -o  heron
$ ./heron
辺1の長さ:
3.0
辺2の長さ:
4.0
辺3の長さ:
5.0
3角形の面積は        6.00です。
$
```