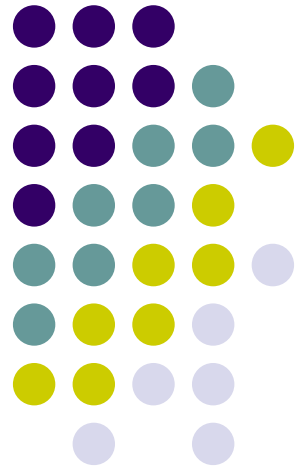
情報処理概論

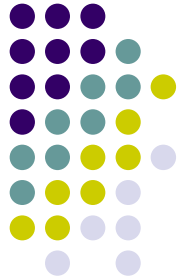
工学部 物質科学工学科
応用化学コース
機能物質化学クラス



第6回

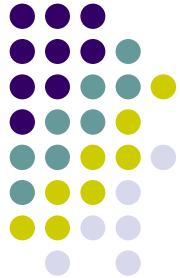
2005年　5月26日

先週の演習2： 円柱の体積



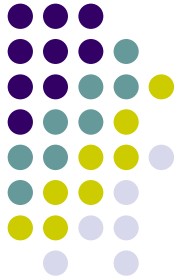
- 半径と高さを入力すると円柱の体積を計算して表示するプログラムを作成する
- read文を利用してキーボードからデータを入力する
- π は atan を利用して生成

先週の演習の正解例



```
program pillar
  implicit none
  real(8) :: r, h, pi
  intrinsic atan

  pi = 4.0D0 * atan(1.0D0)

  write(*, *) 'Takasa: '
  read(*, *) h
  write(*, *) 'Hankei: '
  read(*, *) r

  write(*, *) 'Taiseki = ', h * r * r * pi

stop
end program
```

型の混在

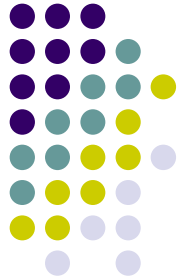


- 整数型と実数型の演算が混在する場合
  型の「強い」方に変換される
  強さ: 倍精度実数＞単精度実数＞整数

- 以下は同じ結果

```
4.0D0 * 3.0D0 / 7.0D0
4.0 * 3.0D0 / 7.0D0
4 * (3.0D0 / 7)
```

プログラム作成の際の注意点

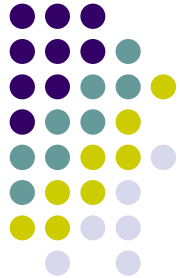


- 型が混在するプログラムを記述する場合
  十分な注意が必要

- 以下は結果が異なる

```
4.0D0 * 3.0D0 / 7.0D0
4.0D0 * (3.0 / 7.0)
4 * 3 / 7
4.0D0 * (3 / 7)
```

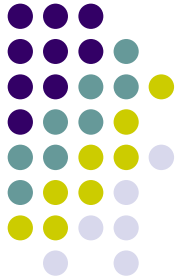
⇒ 用心のため, 型を揃えた方が良い.



○　条件分岐

- 繰り返し
- 型変換

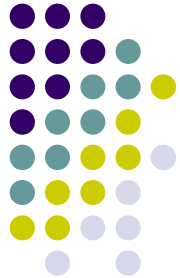
先週の演習で



- 入力された高さや半径が0以下の場合
  入力ミスとして表示し終了するようにしたい.

```
% ./pillar
Takasa: -10.0
 Error: Takasa must be > 0.0
%
```

条件分岐:　if 文の利用（もし～ならば）

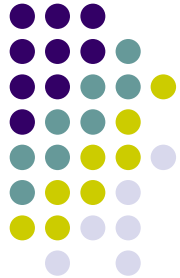


- 条件を満たす場合のみ実行

例)
もし h が　0.0 以下ならば
Error: Takasa must be > 0.0　と表示して終了

```
write(*, *) 'Takasa: '
read(*, *) h
if (h <= 0.0D0) then
  write(*, *) 'Error: Takasa must be > 0.0'
  stop
end if
```

if 文の利用法(1)



- もし～ならば

```
if (条件) then
   条件を満たす場合の処理
end if
```

条件

- if 文で利用できる主な条件

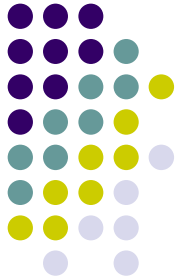
新表記	旧表記	意味
>	.gt.	より大きい
<	.lt.	より小さい
>=	.ge.	以上
<=	.le.	以下
/=	.ne.	等しくない
==	.eq.	等しい

= を2個並べる.
1個だけだと変数への代入の意味になるので注意

if 文の利用例

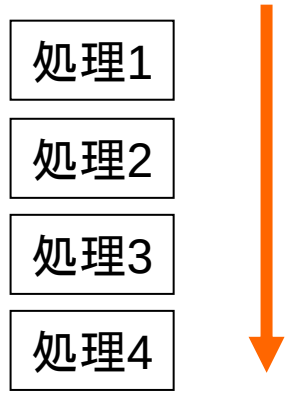
- 降水確率と傘
  - 50%以上なら”傘を持っていきなさい”

```
program pillar
  implicit none
  integer :: prob

  write(*, *) 'Probability of the rain: '
  read(*, *) prob

  if (prob >= 50) then
    write(*, *) "May be it will rain."
    write(*, *) "Don't forget to bring an umbrella!"
  end if

stop
end program
```

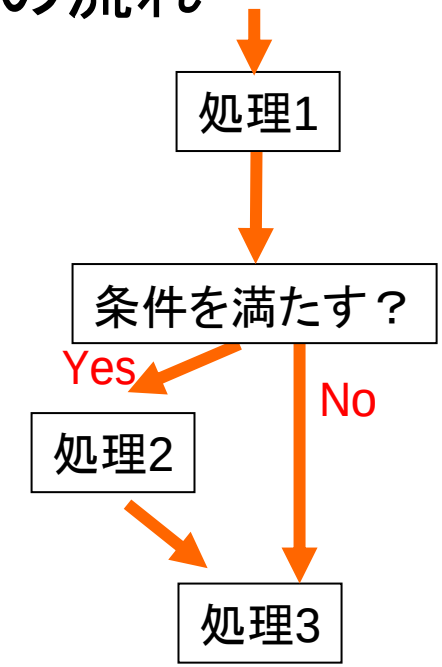


条件分岐

- 先週までのプログラムの流れ
  - 上から順に実行

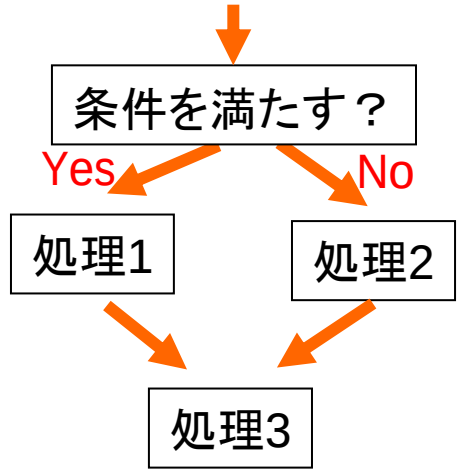


- 条件分岐を含むプログラムの流れ
  - 条件によって処理が変わる

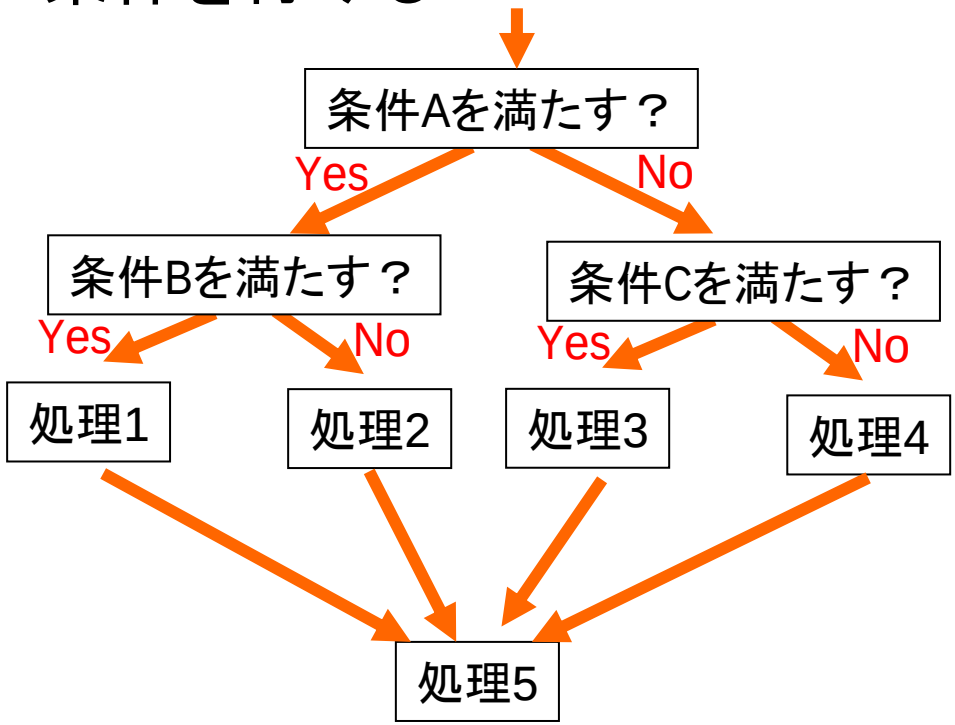


もっと複雑な条件分岐

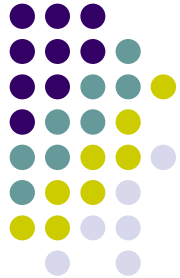
- 条件を満たさない場合の処理



- さらに条件を付ける



条件を満たさなかった場合の処理



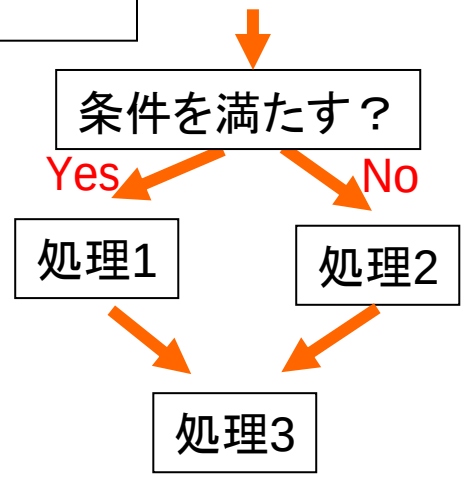
- 50%以上なら”傘を持っていきなさい”
  さもなければ”持っていかなくてもいいかも”

```
if (prob >= 50) then
  write(*, *) "May be it will rain."
  write(*, *) "Don't forget to bring an umbrella!"
else
  write(*, *) "May be it will not rain."
  write(*, *) "You may go without an umbrella."
end if
```

if 文の利用法(2)

- もし～ならば～さもなくば～

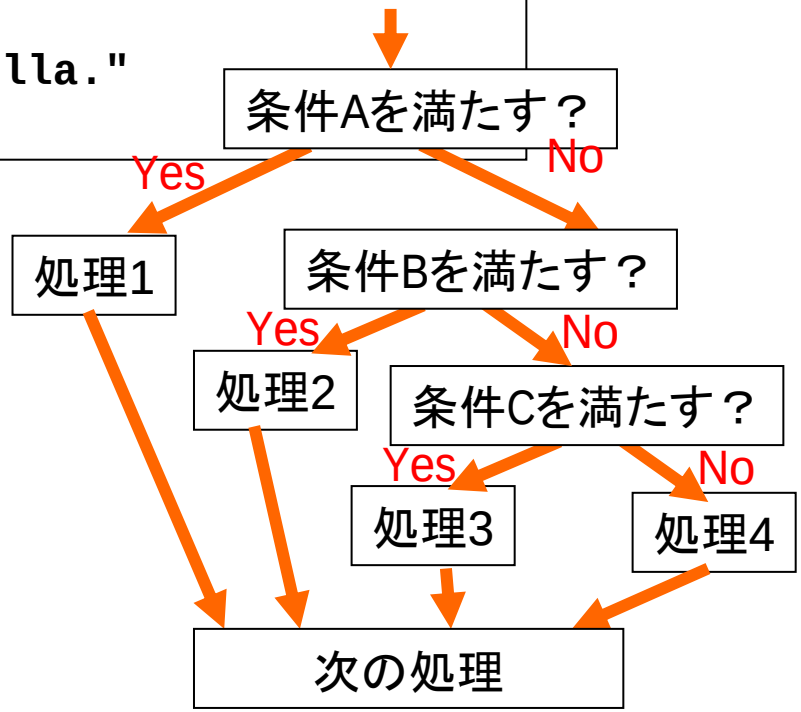
```
if (条件) then
   条件を満たす場合の処理
else
   条件を満たさない場合の処理
end if
```



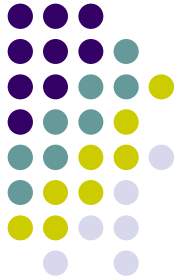
もっと細かく

- 70%以上なら ”持っていきなさい”
  50%以上なら ”持っていったほうがいいよ”
  30%以上なら ”持っていかなくてもいいよ”
  さもなければ ”いらないよ”

```
if (prob >= 70) then
  write(*, *) "Don't forget to bring an umbrella!"
else if (prob >= 50) then
  write(*, *) "You should bring an umbrella."
else if (prob >= 30) then
  write(*, *) "You may go without an umbrella."
else
  write(*, *) "You don't need an umbrella."
end if
```



if 文の利用法(3)



- もし(条件A)ならば～
  さもなくば，もし(条件B)ならば～
  さもなくば，もし(条件C)ならば～
  …
  さもなくば～

```
if (条件A) then
   条件Aを満たす場合の処理
else if (条件B) then
   条件Aを満たさず，条件Bを満たす場合の処理
else if (条件C) then
   条件Aも条件Bも満たさず，条件Cを満たす場合の処理

   ...

else
   どの条件も満たさない場合の処理
end if
```

”または” と ”かつ”

- 条件１　または　条件２

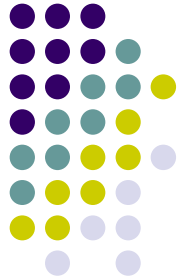
（条件１　.or.　条件2）

- 条件１　かつ　条件２

（条件１　.and.　条件2）

前後のピリオドを忘れずに！

.or. の利用例



- 降水確率が 0未満 または 100より大きい場合エラー

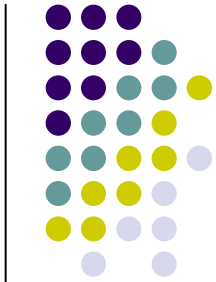
```
if ( prob < 0 .or. prob > 100 ) then
  write(*,*) "Error: Wrong probability ", prob
  stop
else if (prob >= 70) then
  write(*,*) "Don't forget to bring an umbrella!"
else if (prob >= 50) then
  write(*,*) "You should bring an umbrella."
else if (prob >= 30) then
  write(*,*) "You may go without an umbrella."
else
  write(*,*) "You don't need an umbrella."
end if
```

.and. の利用例

- 降水確率が 0未満 または 100より大きい場合エラー

```
if ( prob >= 0 .and. prob <= 100 ) then
  if (prob >= 70) then
    write(*,*) "Don't forget to bring an umbrella!"
  else if (prob >= 50) then
    write(*,*) "You should bring an umbrella."
  else if (prob >= 30) then
    write(*,*) "You may go without an umbrella."
  else
    write(*,*) "You don't need an umbrella."
  end if
else
  write(*,*) "Error: Wrong probability ", prob
  stop
end if
```

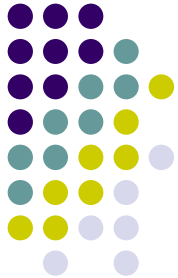
もう一つの方法 select 文



- select 文の利用例

```
select case(prob)
case(70:100)
  write(*, *) "Don't forget to bring an umbrella!"
case(50:69)
  write(*, *) "You should bring an umbrella."
case(30:49)
  write(*, *) "You may go without an umbrella."
case(0:29)
  write(*, *) "You don't need an umbrella."
case default
  write(*, *) "Error: Wrong probability ", prob
  stop
end select
```

select文の利用法



```
select case(式)
case(値1)
  式の値が値1に含まれる場合の処理
case(値2)
  式の値が値2に含まれる場合の処理

   . . .

case default
   式の値がどの値にも含まれない場合の処理
end select
```

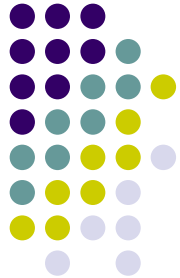
- 式は整数型か文字型か論理型
  - 実数は不可
- どの値にも含まれない場合の処理
  case default は省略可

case で指定できる値

- 数値，文字のみ
  例）20の場合　　case(20)
  　　‘d’ の場合　　case('d')

- 数値や文字列の範囲
  例）0～10の場合　　case(0:10)
  　　‘a’ ~ ‘e’ の場合　　case('a':'e')
  　　50以下の場合　　case(:50)
  　　'n'以降の文字の場合　　case('n':)

- 複数の値
  例）1，5，10 の場合　　case(1,5,10)
  　　‘c‘，’t’ の場合　　case('c','t')



- 条件分岐
- 繰り返し
- 型変換

繰り返し

- 以下のように 2^i (i = 1~10) を表示するプログラムを作るには?

```
2** 1:      2
2** 2:      4
2** 3:      8
2** 4:     16
2** 5:     32
2** 6:     64
2** 7:    128
2** 8:    256
2** 9:    512
2**10:   1024
Done!
```

方法1：全部書く

```
program power_1
  write(*, '(a,i10)') '2** 1:',2**1
  write(*, '(a,i10)') '2** 2:',2**2
  write(*, '(a,i10)') '2** 3:',2**3
  write(*, '(a,i10)') '2** 4:',2**4
  write(*, '(a,i10)') '2** 5:',2**5
  write(*, '(a,i10)') '2** 6:',2**6
  write(*, '(a,i10)') '2** 7:',2**7
  write(*, '(a,i10)') '2** 8:',2**8
  write(*, '(a,i10)') '2** 9:',2**9
  write(*, '(a,i10)') '2**10:',2**10

  write(*, *) 'Done!'
stop
end program
```

- 現実的ではない
  - 数が多くなると大変．例えば　i=1~1000
  - 範囲が変わると全部書き換え．例えば i=20~30

方法2： 繰り返しを使う

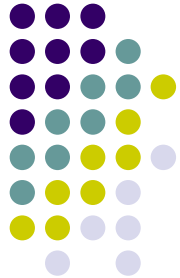
- *i* を 1 から 1つずつ増やしながら10まで繰り返す
  - 繰り返しの内容：

```
write(*, ‘(“2**”, i4, “:”, 1x, i10)’) i, 2**i
```

i と 2^i を表示

```
program power_2
  implicit none
  integer :: i

  do i=1,10
     write(*, '("2**", i4, ":", 1x, i10)') i, 2**i
  end do

  write(*, *) 'Done!'
stop
end program
```

実行イメージ



- プログラム

```
do i = 1, 10
 write(*, '("2**", i4, ":", 1x, i10)') i, 2**i
end do
```

i=1 `write(*, '("2**", i4, ":", 1x, i10)') 1, 2**1`

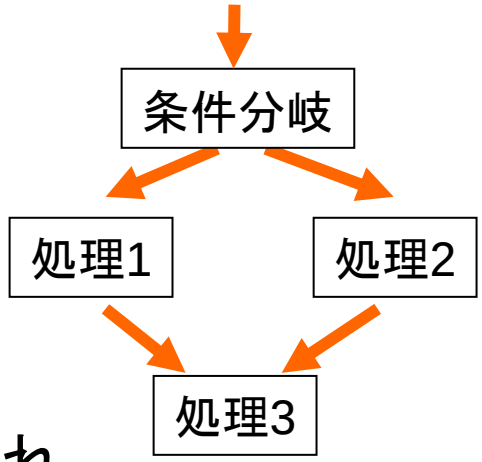
i=2 `write(*, '("2**", i4, ":", 1x, i10)') 1, 2**2`

i=3 `write(*, '("2**", i4, ":", 1x, i10)') 1, 2**3`

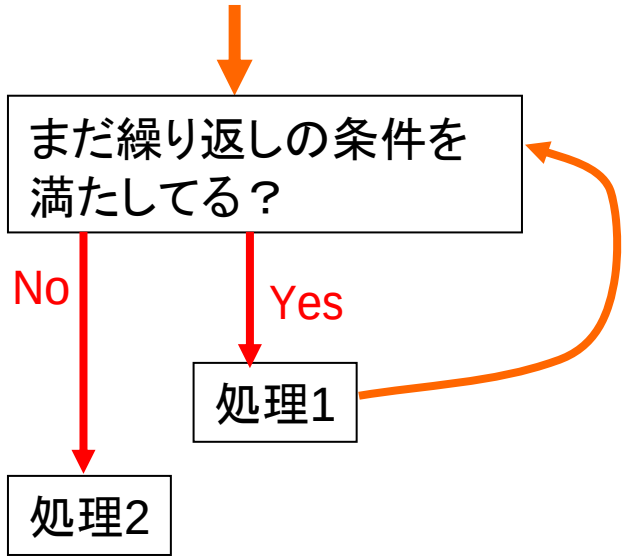
i=10 `write(*, '("2**", i4, ":", 1x, i10)') 1, 2**10`

繰り返し

- 今までのプログラムの流れ
  - 各処理は高々一回実行



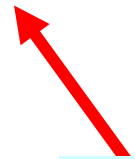
- 繰り返しを含むプログラムの流れ
  - 指定された条件を満たす間
    同じ処理を繰り返して実行する



do文の利用法

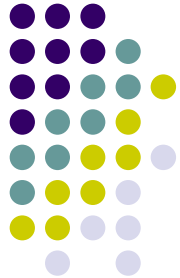
- 制御変数の値を 開始値 から 増分 ずつ
  増やしてゆき，終了値 以上になるまで繰り返す

```
do  制御変数  =  開始値, 終了値  [, 増分]

    繰り返す処理

end  do
```



- 増分を省略すると1が指定されたとみなされる
- 増分は負の値でも良い．
  - この場合，制御変数を開始値から増分ずつ減らしてゆき，
    終了値以下になるまで繰り返す．
    例） do i = 100, 0, -5

開始値，終了値は変数でも構わない



- 例） 開始値,終了値を実行時に入力する.

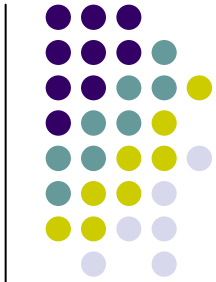
```
program power_3
  implicit none
  integer :: i,start,end

  write(*, '(a$)') 'Start: '
  read(*, *) start
  write(*, '(a$)') 'End: '
  read(*, *) end

  do i=start,end
     write(*, '("2**",i4, ":", 1x, i10)') i, 2**i
  end do

  write(*, *)  'Done!'

stop
end program
```

do文利用上の注意

- do文の繰り返し処理中，制御変数の値を変化させない
  - 途中で値が変化すると繰り返しが正常に終了しない
    - 誤りの例:

```
do i=1,10
   i = 1
   write(*, '("2**",i2, ":", 1x, i5)') i, 2**i
end do
```

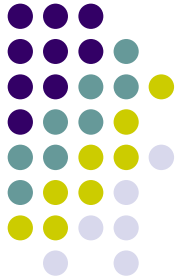
文の組み合わせ



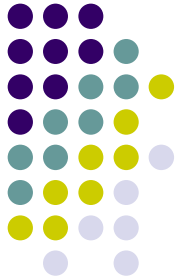
- 実際のプログラムでは if文や do文を組み合わせて記述する
- 注意： do と end do, if と end if は入れ子構造

```
program sample1
implicit none
integer :: i,n

  write(*, *) 'Enter N :'
  read(*, *) n
  if (n <= 0) then
    write(*, *) 'Error: N must be > 0'
  else
    do i = 0, n
      write(*, *) '2**', i, '=', 2**i
    end do
  end if
stop
end program
```

何重に重ねても良い

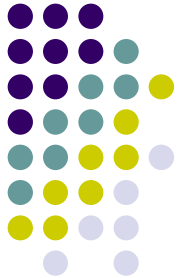


```
program sample1
implicit none
integer :: i, j, n

  write(*, *) 'Enter N :'
  read(*, *) n
  if (n <= 0) then
    write(*, *) 'Error: N must be > 0'
  else
    do i = 1, n
      do j = 1, n
        write(*, *) i, ' * ', j, ' = ', i * j
      end do
    end do
  end if
stop
end program
```



- 条件分岐
- 繰り返し
- ○ 型変換

型の変換： 整数→実数



dble関数 (倍精度実数型へ)

- 制御変数や番号に用いている整数値を実数の計算に利用する場合等

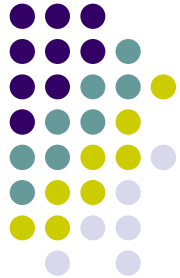
```
program toreal
  implicit none
  integer :: i
  real(8) :: pi
  intrinsic dble,cos,atan

  pi = 4.0D0 * atan(1.0D0)

  do i=1,6
    write(*, *) 'cos(pi/',i,')=', cos(pi/dble(i))
  end do

stop
end program
```

型の変換:　実数→整数

int関数, ceiling関数, nint関数

- 小数点以下の扱いによって使い分け
  - 切り捨て　int
    例）　int(4.2) → 4
  - 切り上げ　ceiling
    例）　ceiling(3.1) → 4
  - 四捨五入　nint
    例）　nint(4.495) → 4

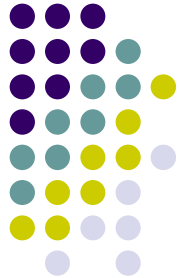
プログラムの字下げの勧め



- プログラムの構造にあわせて字下げすると良い
  - if の条件を満たした場合の処理や
    do文で繰り返される処理など
    → プログラムの相互関係がわかりやすい
  - 例えば，下げ幅は2文字ずつもしくは4文字ずつ

```
if ( prob>=0 .and. prob<=100) then
  if (prob >= 70) then
    write(*, *) "Don't forget to bring an umbrella!"
  else if (prob >= 50) then
    write(*, *) "You should bring an umbrella."
  else if (prob >= 30) then
    write(*, *) "You can go without an umbrella."
  else
    write(*, *) "You don't need an umbrella."
  end if
else
  write(*, *) "Error: Wrong probability ", prob
  stop
end if
```

字下げの有無による見栄えの違い



```
program sample1
implicit none
integer :: i, j, n

write(*, *) 'Enter N :'
read(*, *) n
if (n <= 0) then
write(*, *) 'Error: N must be > 0'
else
do i = 1, n
do j = 1, n
write(*, *) i, ' * ', j, ' = ', i *
end do
end do
end if
stop
end program
```

```
program sample1
implicit none
integer :: i, j, n

  write(*, *) 'Enter N :'
  read(*, *) n
  if (n <= 0) then
    write(*, *) 'Error: N must be > 0'
  else
    do i = 1, n
      do j = 1, n
        write(*, *) i, ' * ', j, ' = ', i * j
      end do
    end do
  end if
stop
end program
```