

情報研究会 C A C T U S

第四回 C言語講習

今回の内容

- ・ 関数
- ・ 変数のスコープ
- ・ 再帰関数

サンプルプログラム

2次方程式の解を調べる プログラム

実行結果

```
1 5 6
2次方程式1x^2+5x+6
異なる2つの実数解を持つ
```

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int discriminant(int,int,int);
5 void check_solution(int);
6
7 int main(void)
8 {
9     int n1, n2, n3, D;
10
11     scanf("%d%d%d", &n1, &n2, &n3);
12     printf("2次方程式%d x^2+%d x+%d\n", n1, n2, n3);
13
14     D = discriminant(n1,n2,n3); //判別式
15
16     check_solution(D); //解の個数を調べる
17
18     return 0;
19 }
20
21 int discriminant(int a, int b, int c)
22 {
23     int d;
24
25     d = pow(b,2) - 4*a*c;
26
27     return d;
28 }
29
30 void check_solution(int d)
31 {
32     if(d>0) printf("異なる2つの実数解を持つ\n");
33
34     else if(d == 0) printf("重解を持つ\n");
35
36     else printf("異なる2つの虚数解を持つ\n");
37 }
```

戻り値のある関数の宣言

```
戻り値の型 関数名 (型 仮引数名, ...)  
{  
    処理  
    return 戻り値  
}
```

実引数 ... 呼び出した関数に渡す値

仮引数 ... 実引数を格納する変数

戻り値 ... 関数を呼び出した側に返す値

戻り値のない関数の宣言

```
void 関数名 (型 仮引数名, ...)  
{  
    処理  
}
```

- ・ 戻り値を返さない関数の型はvoidとなる
- ・ 戻り値がないのでreturn文は書かない

関数の呼び出しかた

関数を呼び出すときは戻り値の型と実引数の型は記述しない

- ・ 戻り値のある関数の呼び出し例

```
D = discriminant(n1, n2, n3);  
printf(“%d”, discriminant(n1, n2, n3));
```

戻り値の型によって代入できる変数や書式指定子の種類が決定される

- ・ 戻り値のない関数の呼び出し例


```
check_solution(D);
```

戻り値のある関数の処理の流れ

```
int D  
D = discriminant(n1, n2, n3)
```



```
a = n1 b = n2 c = n3  
int d  
d = b^2 - 4 × a × c  
return d
```



```
D = d
```

関数を呼び出し、関数に実引数（この場合n1, n2, n3）を渡す

実引数が代入された仮引数を使って処理を実行し、結果を戻り値として返す

戻り値を代入する

確認問題 1

サンプルプログラムに2次方程式が重解を持つときの解を求める関数を追加してください

サンプルプログラム

実行結果

```
a:1 b:1 c:1  
a:3 b:2 c:2  
a:5 b:3 c:3
```

```
1 #include <stdio.h>  
2  
3 void count(int); //関数のプロトタイプ宣言  
4  
5 int a = 0; //グローバル変数  
6  
7 int main(void)  
8 {  
9     int b = 0; //main関数のローカル変数  
10  
11     count(b); a++; b++;  
12     count(b); a++; b++;  
13     count(b); a++; b++;  
14  
15     return 0;  
16 }  
17  
18 void count(int n)  
19 {  
20     static int c = 0; //静的ローカル変数  
21  
22     n++;  
23     a++;  
24     c++;  
25  
26     printf("a:%d b:%d c:%d\n", a, n, c);  
27 }
```

変数のスコープ

変数には使用できる有効範囲があり、これを変数のスコープという

グローバル変数…スコープがプログラム全体である変数

ローカル変数…スコープが変数を宣言したブロック内である変数

グローバル変数

- ・ 関数外で宣言された変数をグローバル変数という
- ・ グローバル変数は宣言した場所より下に記述されたすべての関数で利用できる

使用上の注意点

グローバル変数の中身はどの関数からでも書き換えることができるので、デバッグやコードリーディングが面倒になる

ローカル変数

- ・ 関数内で宣言された変数をローカル変数という
- ・ ローカル変数は宣言した関数内でしか使用できない
- ・ 関数の処理が終了するとメモリ上から解放される（自動変数）

※別々の関数で同じ変数名を宣言してもそれらは異なる変数である

関数1

int a;

≠

関数2

int a;

静的ローカル変数

- ・ 変数を宣言するときに型名の前に`static`を付ける
- ・ 静的ローカル変数は関数の処理が終了してもメモリ上から解放されない
- ・ 関数が最初に呼び出されたときの一度だけ初期化される
- ・ 初期化を行わない場合は自動的に初期値が0になる

サンプルプログラム

nの階乗を求めるプログラム

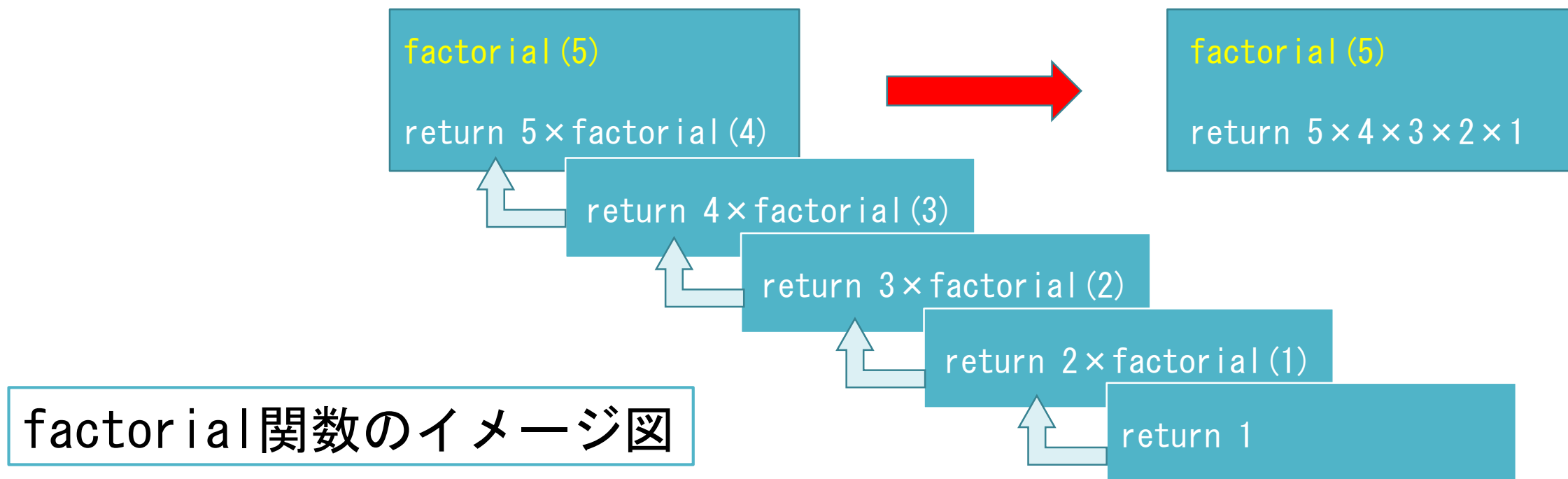
実行結果

```
数字を入力してください:5  
呼び出し1回  
呼び出し2回  
呼び出し3回  
呼び出し4回  
呼び出し5回  
5の階乗は120
```

```
1 #include <stdio.h>  
2  
3 int factorial(int);  
4  
5 int main(void)  
6 {  
7     int n;  
8  
9     printf("数字を入力してください:");  
10    scanf("%d", &n);  
11  
12    printf("%dの階乗は%d\n", n, factorial(n));  
13  
14    return 0;  
15 }  
16  
17 int factorial(int n)  
18 {  
19     //関数の呼び出し回数を表示  
20     static int count = 1;  
21     printf("呼び出し%d回\n", count);  
22     count++;  
23  
24     //階乗を計算  
25     if(n==1) return 1;  
26  
27     return n*factorial(n-1);  
28 }
```

再帰関数

再帰呼び出しをする関数を**再帰関数**という
再帰呼び出しとは関数の処理で自身の関数を呼び出すことである



確認問題 2

サンプルプログラムに 1 から n までの総和を求める再帰関数を追加してください

練習問題 1

整数を5つ入力してそれらを半径とした球の体積と表面積を求めるプログラム

ただし円周率は3.14、有効数字を小数点第2位までとする

実行例

```
1 5 2 9 0
r:1
体積:4.19 表面積:12.56
r:5
体積:523.33 表面積:314.00
r:2
体積:33.49 表面積:50.24
r:9
体積:3052.08 表面積:1017.36
r:0
体積:0.00 表面積:0.00
```

練習問題 2

再帰関数でフィボナッチ数列の第 n 項を求めるプログラム

(参考) フィボナッチ数列
 $f(n) = f(n-1) + f(n-2)$
 $f(2) = 1$
 $f(1) = 1$

1 1 2 3 5 8 13 21 ...

実行例

```
n:8  
F(n):21
```

練習問題 3

学籍番号のチェックディジットを求めるプログラム

(参考) チェックディジットの求め方

1. 学籍番号の各桁に2から7を掛けて足し合わせる
2. 足し合わせた数を11で割った余りに対応する
アルファベットを変換表から探す

例) 152910の場合

$$1 \times 7 + 5 \times 6 + 2 \times 5 + 9 \times 4 + 1 \times 3 + 0 \times 2 = 86 \quad 86 \% 11 = 9 \quad 9 \rightarrow C$$

変換表

0	→	A
1	→	A
2	→	Z
3	→	Y
4	→	X
5	→	U
6	→	M
7	→	K
8	→	H
9	→	C
10	→	B

学籍番号
152910C