知能情報工学演習I
第12回（後半第6回）
課題の回答

岩村雅一

masa@cs.osakafu-u.ac.jp

前回の課題1

- 球の体積を計算するマクロを作り、球の半径(小数とする)を入力したとき、球の体積を返すプログラムを作成しなさい。

```
#include<math.h>
#include<stdio.h>
#define V(r) 4.0/3.0*M_PI*r*r*r

int main(void){
  float i;
  printf("半径を小数で入力：");
  scanf("%f",&i);

  printf("半径%fの球の体積は%fです。
     ¥n",i,V(i));

  return(0);
}
```

マクロ中の
rをiに置き換える

課題1で実際にあった間違い（その1）

- 球の体積を計算するマクロを作っていない。（題意に反する）
- 係数が整数扱い（4/3=1）になっている

  #define V(r) 4/3*M_PI*r*r*r
- 体積の公式を間違っている。
  - 3/4 π r^3

課題1で実際にあった間違い（その2）

- マクロの使い方が正しくない。
  （たまたま動いた）

```
#define ts(r) 4*M_PI*a*a*a/3
…
printf("その球の体積は%f¥n",ts(a));
```

rをaに置き換えようとしても、
もともとrがない。
今回はたまたまaを
使っていたので、うまくいった。

もし、呼び出し側でa以外の変数を使っていたら、正しく動かなかった。

- 間違いではないが、ファイル名がおかしいため、危うく見落とすところだった。
  - 0710-10801070**-1.c

前回の課題2

- 階乗(1からnまでの自然数の積)を計算する関数を作り、順列と組み合わせを表示しなさい
  - 順列

$$ {}_nP_r = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!} $$

  - 組み合わせ

$$ {}_nC_r = \frac{{}_nP_r}{{}_rP_r} = \frac{n!}{r!(n-r)!} $$

前回の課題の回答例1 (for文を使った場合)

```
#include <stdio.h>

/* 階乗を計算する関数 */
int fact(int x) {
  int i, fact = 1;
  for(i = 2; i <= x; i++) {
    fact *= i;
  }
  return(fact);
}
```

fact = 1 * 2 * 3 * 4 *…

0!や1!もok

```
int main (void){
  int n, r, p, c;

  printf("n: "); scanf("%d", &n);
  printf("r: "); scanf("%d", &r);

  p = fact(n) / fact(n-r);
  c = fact(n) / fact(n-r) / fact(r);

  printf("nPr = %d¥n", p);
  printf("nCr = %d¥n", c);

  return(0);
}
```

前回の課題の回答例2（関数の再帰的呼び出し）

main関数は説明のために消去

```
#include <stdio.h>

/* 階乗を計算する関数 */
int fact(int x) {
  if (x==1 || x==0) {
    return(1);
  } else {
    return(x*fact(x-1));
  }
}
```

```
int main (void){
  int n, r, p, c;

  printf("n: "); scanf("%d", &n);
  printf("r: "); scanf("%d", &r);

  p = fact(n) / fact(n-r);
  c = fact(n) / fact(n-r) / fact(r);

  printf("nPr = %d¥n", p);
  printf("nCr = %d¥n", c);

  return(0);
}
```

前回の課題の回答例2（関数の再帰的呼び出し）

例：fact(2)の場合

```
#include <stdio.h>

/* 階乗を計算する関数 */
int fact(int x) {
  if (x==1 || x==0) {
    return(1);
  } else {
    return(x*fact(x-1));
  }
}
```

fact(2)
=
fact(1)
=
1

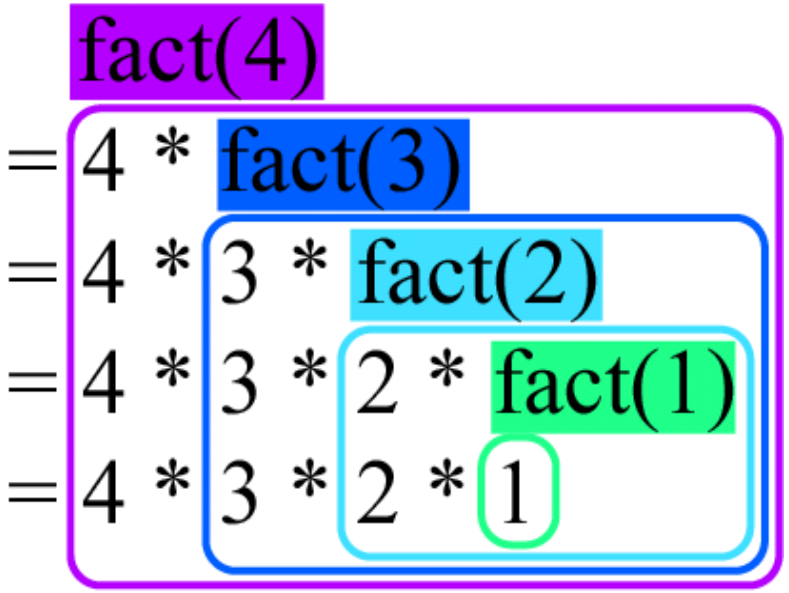
fact(1)の計算

```
/* 階乗を計算する関数 */
int fact(int 1) {
  if (x==1 || x==0) {
    return(1);
  } else {
    return(1*fact(1-1));
  }
}
```

前回の課題の回答例2
（関数の再帰的呼び出し）

```
#include <stdio.h>

/* 階乗を計算する関数 */
int fact(int x) {
  if (x==1 || x==0) {
    return(1);
  } else {
    return(x*fact(x-1));
  }
}
```



課題2で実際にあった間違い（その1）

- コンパイルが通らない。
- 階乗を計算する関数を作っていない。
  （題意に反する）
- 計算式が違う。
  - 順列? $${}_nP_r = \frac{n!}{r!}$$

正解は
$${}_nP_r = \frac{n!}{(n-r)!}$$

課題2で実際にあった間違い（その1）

- コンパイルが通らない。
- 階乗を計算する関数を作っていない。
  （題意に反する）
- 計算式が違う。
  - 順列? $${}_nP_r = \frac{n!}{r!}$$

正解は
$${}_nP_r = \frac{n!}{(n-r)!}$$

課題2で実際にあった間違い（その2）

- main関数の中で関数を定義している（関数の入れ子）。
  - GNU Cコンパイラの拡張機能なので、使えない環境がある。

```
int main(void){

  int bikkuri(x){
      …
    return t;
  }

  int n,r,nPr,nCr;
```

今回は正解にしました