C言語講座第3回 2018

関数とポインタ

前回の演習３解答例

```
#include<stdio.h>
#include<string.h>
#define MAX 20
int main(void)
{
  int i, j, k, l, n;
  int num = 7;
  char moji[7][MAX], tmp[MAX];
  printf("¥n◆7つの文字列（半角英数字）を入力してください¥n");
  for (i=0;i<num;i++) {
    printf("%dつめの文字列 = ", i+1);
    scanf("%s", moji[i]);
  }
  for(i=1;i<num;i++){
    for(j=1;j<num;j++){
      for(k=0;k<MAX;k++){
        }
for(l=0,n=k;l<k;l++){
          if(moji[j-1][l] == moji[j][l])n--;

 }
        if(n == 0 && moji[j-1][k] > moji[j][k]){
          strcpy(tmp, moji[j-1]);
          strcpy(moji[j-1], moji[j]);
          strcpy(moji[j], tmp);
        }
      }
    }
  }
  printf("¥n◆文字列をソートしました。¥n");
  for (i=0;i<num;i++) printf("%s¥n", moji[i]);
  return 0;
}
```

初めは動けば良い

問：小文字はaからzまでの26文字である。では乱数を用い'a'または'z'だった時、出力するプログラムを作成せよ。

```
random = 'a' + rand() % ('z' - 'a' + 1);
if( (random == 'a' ) || (random == 'z') )
{
    printf("%c",random);
}
```

```
random = rand();
if( (random == 'a' ) || (random == 'z') )
{
    printf("%c",random);
}
```

上の二つのプログラムの違いって何？

マクロ定義

```
#define P 3+5+7
#define Q 3*P

//Q=3*3+5+7=21
```

```
#define P (3+5+7)
#define Q 3*P

//Q=3*(3+5+7)=45
```

defineは多重使用できる。
また、置換後の式全体をかっこで囲まないと予想外の挙動になる場合がある。

関数とは

複数の処理をまとめたもの。
一連の処理をくり返し使いたい時などに便利。

stdlib.hの中のrand()など、汎用性の高いものはあらかじめ用意されている。

今回はこの関数を自作する。

関数を使わないと

繰り返したい処理を
main()の中に何度も
書かなければならない。

1つの間違いを修正するために
全ての箇所をチェックするはめになる。

```
#include <stdio.h>

int main(){
	char a;

	scanf("%c", &a);
	printf("%c", a);

	scanf("%c", &a);
	printf("%c", a);

	return 0;
}
```

関数を使うと

詳しい解説は次ページ以降で行うが、
処理の中身を書くのは
一度だけで良くなる。

修正が楽になる他、
main()の構成が分かりやすくなる等の
利点がある。

```
#include <stdio.h>
void mojiprint(){
        char a;
        scanf("%c", &a);
        printf("%c", a);
}
int main(){

        mojiprint();
        mojiprint();

        return 0;
}
```

関数の呼び出し

ステップ1：一番簡単な関数

```
void mojiprint(){
	//処理
}
```

関数の名前は任意。

後述のプロトタイプ宣言をしない場合は

自作関数を呼び出す前に処理の中身を書いておく必要がある。

(先ほどの使用例でいえばmain関数の前にmojiprint関数が書いてある)

演習1

自分の名前を出力する関数を作成し、
それをmain関数から呼び出すプログラムを作成せよ。

演習1解答例

printfはstdio.hの中にあるため、
pnの前にincludeする必要がある。

```
#include <stdio.h>
void pn(){
        printf(”my name¥n");
}
int main(){
        pn();
        return 0;
}
```

プロトタイプ宣言

自作関数の並び順が
上手く定まらない場合、
プロトタイプ宣言を行う。

右の例のpn();のように、
関数名();と書く。
これをすることで
呼び出し先の関数が
下にあっても呼び出すことができる。

```
#include <stdio.h>
void pn();
int main(){
        pn();
        return 0;
}
void pn(){
        printf(”my name¥n");
}
```

ステップ2：引数(ひきすう)

```
void printnum(int n){
        printf("%d¥n", n);
}
```

関数名の後の()で変数を宣言すると

その関数に値を渡すことができる。

引数使用例1

printnumで宣言しているnに
3を代入して処理を行う。

```
#include <stdio.h>
void printnum(int n);
int main(){
	printnum(3);

	return 0;
}
void printnum(int n){
	printf("%d¥n", n);
}
```

引数使用例2

引数をカンマで区切って
複数指定することもできる。

右の例では第一引数のprenの値を
printsumの第一引数nに代入している。

prerとrについても同様。

```
#include <stdio.h>
void printsum(int n, double r);
int main(){
	int pren = 3;
	double prer = 4.1;
	printnum(pren, prer);

	return 0;
}
void printsum(int n, double r){
	printf("%f¥n", n+r);
}
```

ステップ3：戻り値

```
int calcsum(int m, int n){
        return m+n;
}
```

関数名の前に戻り値の型を書いた場合、

returnを使うことで

その型の値を返すことができる。

先ほどまで書いていたvoidは戻り値が無いことを示す。

戻り値のことを返り値ともいう。

戻り値使用例

calcsum(int, int)がその値に変わると考えればよい。

右の例ではsumにcalcsumの戻り値を代入している。

```
#include <stdio.h>
int calcsum(int m, int n);
int main(){
	int prem = 3;
	int pren = 4;
	int sum;
	sum = calcnum(prem, pren);
	printf("%d¥n", sum);
	return 0;
}
int calcsum(int m, int n){
	return m+n;
}
```

演習2

int型変数prea, prebにscanfで値を入力したのち、

その二つの値の差(preb - prea)を返す関数を呼び出しなさい。

また差が、正である場合、等しい場合、負である場合、それぞれ異なる文字列を出力せよ。

演習2解答例

```
#include <stdio.h>
int calcdist(int a, int b);
int main(){
        int prea, preb,dist;
        scanf("%d", prea);
        scanf("%d", preb);
        dist = calcdist(prea, preb);
        if(dist > 0) printf(“正”);
        else if(dist == 0) printf(“等しい”);
        else printf(“負”);
        return 0;
}
```

```
int calcdist(int a, int b){
        return b-a;
}
```

関数内での変数宣言

関数では引数以外にも
変数を宣言できる。
関数内で宣言した変数は
その関数内でのみ使用可能で、
main関数内では使用不可能。

例でいうとnumという変数はaddnumの
実行中のみ存在しており、
addnumの終了時に破棄。(未定義状態)

```
#include <stdio.h>
int addnum(int a);
int main(){
	int prea, a2;
	scanf("%d", prea);
	a2 = addnum(prea);
	printf("%d", a2);
	return 0;
}
int addnum(int a){
	int num;
	scanf("%d", &num);
	return a + num;
}
```

変数のスコープ

逆に、main関数で宣言した変数を
直接他の関数で使うことはできない。

右の例ではmainとato5のどちらにも
aという変数があるが、
これらは同じ名前の別物であり、
ato5側でaを5に変えても
mainのaは変わらず3のままである。
変数の有効範囲のことを
その変数のスコープという。

```
#include <stdio.h>
void ato5(int a);
int main(){
	int a = 3;
	ato5(a);
	printf("%d", a);//3

	return 0;
}
void ato5(int a){
	a = 5;
}
```

間接代入

main関数の変数を変更する方法として
ポインタを使う方法がある。

右の例ではato5を呼び出す時
引数を&aにしている。
これによってato5(int *pa)の
paに&aが代入され
*paがaを指すようになっている。

```
#include <stdio.h>
void ato5(int a);
int main(){
	int a = 3;
	ato5(&a);
	printf("%d", &a);//5

	return 0;
}
void ato5(int *pa){
	*pa = 5;
}
```

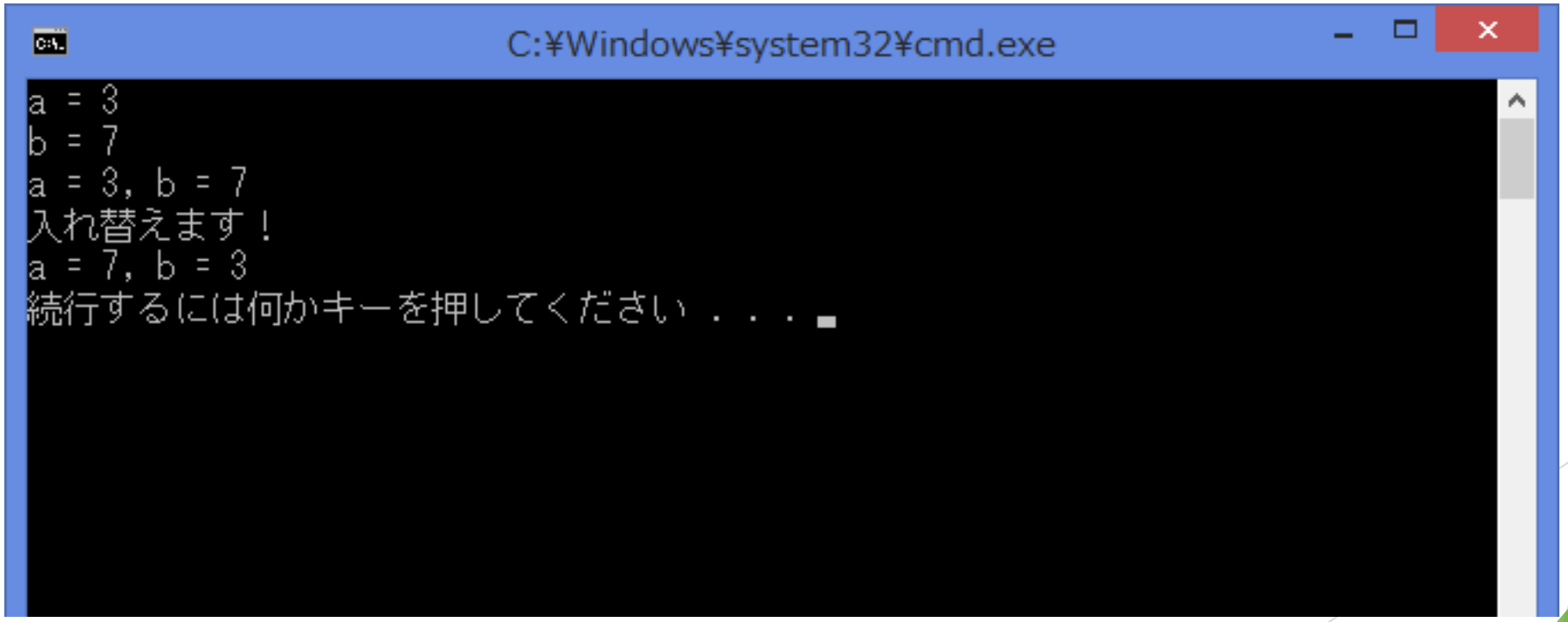
int *pa = &a
ポインタ変数の
宣言時代入は
pa = &a

演習その3

int型変数a,bをscanf関数を用いて入力させて出力したのち、

自作関数swap()で値a, bを交換し、

main関数でa,bを出力しなおしてください。



演習その3解答例

```
#include <stdio.h>
void swap(int *x, int *y){
        int tmp;
        printf("入れ替えます！¥n");
        tmp = *x;
        *x = *y;
        *y = tmp;
}
```

```
int main(void){
        int a, b;
        printf("a = ");
        scanf("%d", &a);
        printf("b = ");
        scanf("%d", &b);
        printf("a = %d, b = %d¥n", a, b);
        swap(&a, &b);
        printf("a = %d, b = %d¥n", a, b);

        return 0;
}
```

文字列の導入

C言語では文字列をcharの配列で表す。

char a[5]={'E', '1', '0', '1', '¥0'};とすれば、"E101"という文字列を作れる。

ここで'¥0'とは、その文字列が終了することを表すNULL文字であり、

文字列の最後に必ず入れなければならない。

char a[5]="E101";やchar a[]="E101"と書くこともできるが、

この場合も5文字分用意しなければならないことに注意。

文字列の入出力

文字列を入出力したい場合、"%s"(string)という指定子が使われる。
例えばchar a[5]="E101"を表示したい場合は、
printf("%s", a);とすればよい。
ここでのaはa[0]のアドレスであり、
これを利用してa[0]、a[1]、…と順に表示していく。

逆に入力したい場合はa[5]が宣言してある状態で
scanf("%s", a);とすればよい。

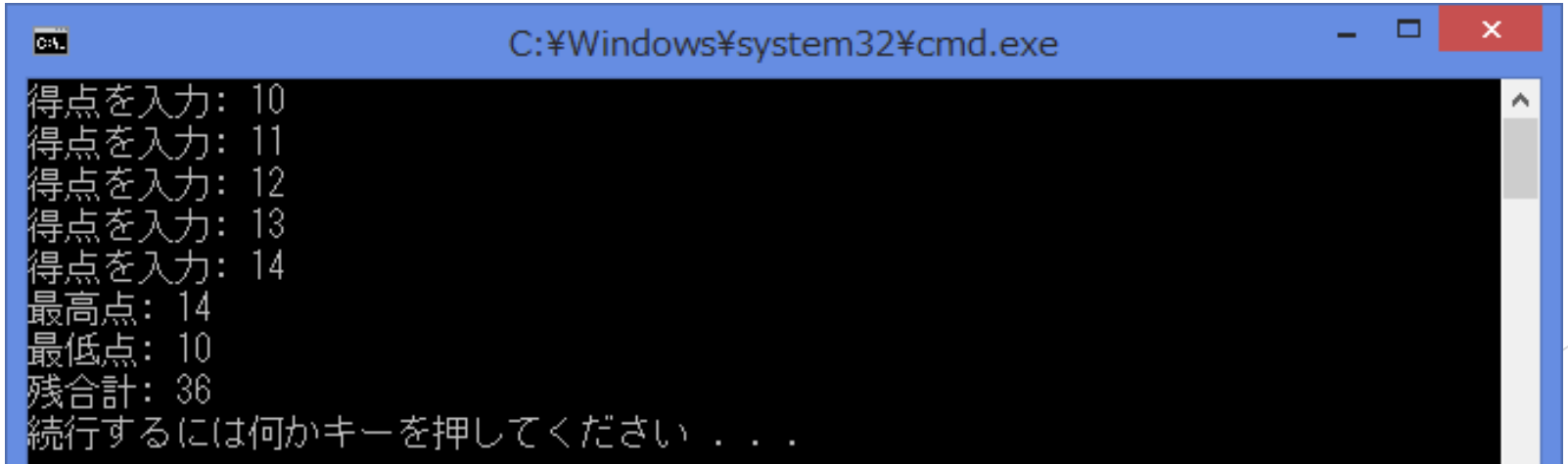
scanf("%s", a);の注意点

scanf("%s", a);をそのまま使うのは危険性が高いのですが、
授業で深く扱わないのでここでも深く扱いません。

ここで最低限覚えておくべきなのは、
・空白文字の入力は禁止
・a[5]に対して入るのは4文字だけ
の2つです。
名前などを入力させる際は、
予想される文字列の長さより大きめの配列を宣言しておきましょう。

演習その4

5名のそれぞれの得点を0～20点の範囲で手入力させた後、
最高点と最低点の各1名を除いた、残り3名の得点の合計を求めてください。
ただし、
2つの数値のうち小さい方の値を戻り値とする関数lt(a,b)
2つの数値のうち大きい方の値を戻り値とする関数gt(a,b)
を作成し、使用してください。



演習その4 解答例

```
#include <stdio.h>

int lt(int a, int b);
int gt(int a, int b);

int main(){
        int pt, i;
        int max = 0;
        int min = 20;
        int sum = 0;
        for (i = 0; i < 5; i++){
                printf("得点を入力: ");
                scanf("%d", &pt);
                max = gt(pt, max);
                min = lt(pt, min);
                sum += pt;
        }
        printf("最高点: %d¥n", max);
        printf("最低点: %d¥n", min);
        printf("残合計: %d¥n", sum - (max + min));
        return 0;
}
```

```
int gt(int a, int b){
        if(a > b) return a;
        else return b;
}
int lt(int a, int b){
        f(a < b) return a;
        else return b;
}
```

補足

関数やNULL文字に関して

main関数

main関数自体も関数であるが、

mainは「特別な名前」として定義されており、

プログラム実行時に最初に呼び出されることになっている。

プロトタイプ宣言の略記法

printnum(int);

のように、変数名を省略することができる。

特殊なスコープ1:グローバル変数

main関数を含むどの関数にも属していない部分でも
変数を宣言することができる。
この変数はグローバル変数と呼ばれ
どの関数からも参照および変更が可能。
ただしその分危険性も高いため、安易には使えない。

特殊なスコープ2:静的(static)変数

(主に自作の)関数内での変数宣言で

static int a;

のように前にstaticを付けると

その変数は関数が終了しても保持される。

ただし、その関数からしか参照および変更ができない。

次ページに使用例を示す。

静的変数使用例

右の例で

static int i = 0;

によるiの宣言と初期化は

1回しか行われない。

また、このiをcount以外の関数から

参照、変更することはできない。

```
void count();
int main(){
	count();//1回目の呼び出し
	count();//2回目の呼び出し
	return 0;
}
void count(){
	static int i = 0;
	i++;
	printf("%d回目の呼び出し", i);
}
```

特殊なスコープ3:for内限定

for(int i=0; i<N; i++){}
のようにforの初期化式内で変数を定義すると、
その変数は当該for内でしか参照、変更ができない。
(for文終了時に破棄される。)

スコープの意味

右のswap(int*, int*)は
演習その3で使ったものだが、
これをmain関数で書くと
tmpという
今後一切使わない変数も
保持され続ける。

スコープの限定は
間違った参照、代入を防ぐ以外に
保持する変数を減らして
実行速度を上げる意味もある。

```
#include <stdio.h>
void swap(int *x, int *y){
        int tmp;
        printf("入れ替えます！¥n");
        tmp = *x;
        *x = *y;
        *y = tmp;
}
```

"%s"とNULL文字

char a[5]={'E', '1', '0', '1', '¥0'};を宣言したうえで、
printf("%s", a);とすると、"E101"と表示される。

これに対し、
char a[5]={'E', '1', '¥0', '1', '¥0'};と宣言したうえで、
printf("%s", a);とすると、"E1"と表示される。
これはprintf("%s", a);アドレスaに入っている文字から1文字ずつ
NULL文字が出てくるまで表示していく命令だからである。

再帰関数(recursion)

右のzenka(int n)は
引数nに対し
・$a_1=3$
・$a_n=5a_{n-1}+2(n\geqq 2)$
を返す。

この例では
一般項を求めれば済んでしまうが
もっと問題が難しい場合や
ランダムアクセス不可能な
データを扱う場合に使う。

```
#include <stdio.h>
int zenka(int n){
	if(n == 1){
		return 3;
	}
	else{
		return 5*zenka(n-1)+2;
	}
}
int main(){
	printf("%d", zenka(4));
	return 0;
}
```

演習その5

二人の名前をそれぞれscanfで取得し、昇順にソートして出力するプログラムを作成せよ。取得する名前は全て小文字であると想定して良い。ただし二人の名前を比較する関数を作成し利用せよ。

余裕のある人は大文字の入力も想定し、大文字と小文字の区別のないプログラムを作成せよ。

```
一人目の名前を入力して下さい：alisia
二人目の名前を入力して下さい：alice
名前の若い順に出力します
alice,alisia
```

```
一人目の名前を入力して下さい：Alisia
二人目の名前を入力して下さい：alice
名前の若い順に出力します
alice,Alisia
```

お疲れさまでした！