ソフトゼミＡ　第６回　関数

今まで、printf や scanf といった。関数を使ってきました。これらだけでも十分便利（というより、これらを自分で解決するのは難しい）なのですが、関数を自作することによって、より分かりやすくソースを書けます。今回は関数の作り方と使い方を学びましょう。

関数とは？

関数とはほかの関数（main 文も含む）内で呼び出すことによって、定義した手順を行うものです。たとえば printf は≪二重引用符の中の文字列を%d や%f などの書式指定に従って画面に表示する≫といった具合にです。

関数の定義

関数を使うためにはまずどんな手順を行わせるかという定義をしなければいけません。
関数の定義は必ずほかの関数の外でやります。
関数の定義は次のような形式で書きます。

```
①型 ②関数名( ③引数の型１ 引数１, 引数の型２ 引数２,… ){
 ④内容
}
```

この形式は実は main 文も同じです。つまり、main 文も関数の一つです。（他との差はプログラム実行の際に最初に呼び出すことが約束されていることです。）また、ほかの関数も基本的に main 文と同じ書き方をします。

例

```
#include <stdio.h>
int main(void){
        (main 文の手順)
}
```

（次ページ）

```
//関数の定義
void name(void){
        (関数 name の手順)
}
```

ここでは型は void、関数名は name、引数は無し（void は無しという意味）となってます。これらの詳しい説明は後にします。

関数の使い方

ある関数をほかの関数内で呼び出したいときは呼び出したい関数名と引数を書きます。ここで注意すべきなのは、コンパイラはソースの上から順次処理し、定義してない関数がほかで使われていると型が int 型であると仮定しコンパイルするのでエラーが起きる場合があります。なので、使いたい関数を呼び出す関数の前に定義するか、もしくは呼び出す関数の前に後でどんな関数が定義されるか宣言しないといけません。この前に書く宣言をプロトタイプ宣言といいます。書き方は定義の引数部分まで書き、セミコロンをつけたものです。

例）

```
void name(void);
```

実際に 1 から 10 までの合計を求めて出力する関数 add を定義し、使ってみましょう。

a06_1.c

```
#include <stdio.h>
void add(void);
int main(void){
    int sum = 0;
    add();
    printf("%d¥n",sum); // main 関数の sum は add 関数の sum とは全くの別物！
    return 0;
}
//関数の定義
void add(void){
    int sum = 0,i;
    for(i = 1;i <= 10;i++){sum += i;}
    printf("%d¥n",sum)
}
```

と書きます。ここで注意すべき点としては関数内で定義した変数はほかの関数内で使えないということです。つまり先ほどの例では main 文の中では i という変数は使えず、また sum も main 文内のそれとは全くの別物であります。従って、add 関数にある printf 関数は add 関数の sum の値(1 から 10 までの整数の総和)の 55 を、main 文にある printf 関数は main 関数の sum の値の「0」を吐き出します。

値の受け渡し

関数たちの間で変数が共有されないことを説明しましたが関数間で数値を受け渡しできないと何かと不便です。前の例でいうと add は≪1 から 10 まで≫という限られた区間でしか合計を求められません。それを解消するために引数と戻り値というものがあります。

- 引数とは？
  呼び出し元の関数から呼び出した関数に数値を渡す仕組み。書き方自体は１ページ目参照。呼び出し元の関数のほうを実引数といい、呼び出される関数のほうを仮引数といいます。
- 戻り値とは？
  関数の結果として呼び出し元に返す値。return とセミコロンの間に返す値を書きます。関数の定義の時に書く型はこの戻り値の型です。（戻り値がないときは void 型で定義する。）

これだけの説明だけだと分かりづらいので値を返す場合の例として先ほどの 1 から 10 までの合計を出すプログラムを入力した範囲の合計を出すプログラムに改造してみましょう。

a06_2.c

```
#include <stdio.h>
int add(int a,int b);
int main(void){
    int a,b,sum;
    scanf("%d",&a);scanf( "%d" ,&b);//a は b 以下とする。
    if(a > b){printf("error¥n");return 0;}
    sum = add(a,b);
    printf("%d¥n",sum);
    return 0;
}
```

（次ページ）

```
//関数の定義
int add(int a,int b){
    int sum = 0,i;
    for(i = a;i <= b;i++){sum += i;}
    return sum;
}
```

このプログラムの動作をトレースすると
①a と b に値を入力する。(a が b より大きければエラー表示してプログラムを終了する。)
②add 関数を呼び出し、add 内の変数 a,b に main 文内の a,b の値をそれぞれコピーする。
③add 内の sum に a から b までの数値が全部足される。
④return によって sum の値が関数の戻り値として返される。
⑤返された値が main 文内の sum に代入される。
⑥main 文内の sum が出力される。
となります。

値を返さない場合の例として 1,3～9 の数字をその数値分だけ出力するプログラムを作りましょう。

a06_3.c

```
#include <stdio.h>
void print_num(int max);
int main(void){
    int i;
    for(i = 1;i <= 9;i++){print_num(i);}
    return 0;
}
void print_num(int max){
    int i;
    if(max == 2){return;}
    for(i = 0;i < max;i++){printf("%d",max);}
    printf("¥n");
    return;
}
```

※void 型関数は関数の一番下の部分まで来ると勝手に return するので最後の部分の return 文は省略可

このプログラムをトレース(実際行われるプロセスの確認)すると
①main 文から print_num に 1 を渡して呼び出す。
②print_num が 1 を一回出力する。
③main 文から print_num に 2 を渡して呼び出す。
④if 文により何もせず print_num が終了する。
①main 文から print_num に 3 を渡して呼び出す。
②print_num が 3 を一回出力する。

以下略
ここで仮引数は実引数をコピーしたものであり、仮引数を変更しても実引数となった変数には変化はありません。また、戻り値の方法だと function 内の数値を一つしか返せませんが、このほかにポインタを使う方法があります。(詳しくは次回やります。)

グローバル変数

先ほど、ある関数で定義された変数は、ほかの関数では使えないと書きました。このような各関数の中でしか使えない変数を“ローカル変数”といいます。これに対して、ソース内の全ての関数で使える変数を“グローバル変数”といいます。定義の仕方は関数外でローカル変数同様に定義します。

a06_4.c

```
#include <stdio.h>
int x,y;
void move(int dx,int dy){
    x += dx;
    y += dy;
}
void print_pos(void){
    printf("(%d,%d)¥n",x,y);
}
int main(void){
    int dx,dy;
    x = 0;
    y = 0;
```

（次ページ）

```
    while(1){
        scanf("%d%d",&dx,&dy);
        if(dx == 0 && dy == 0){break;}
        move(dx,dy);
        print_pos();
    }
    return 0;
}
```

グローバル変数は初期化しなくても値が 0 になるので main 文の初期化部分を消しても問題なく動きます。また関数内でグローバル変数と同じ名前のローカル変数を定義した場合ローカル変数が優先されます。(しかし、分かりづらいので名前は別々のほうがいいです。)また、グローバル変数はどこからでも書き換えができてしまうのでバグを引き起こしやすく、発見も難しいのでできるだけ使用は控えるべきです。

練習問題

次の関数を作成してください。main 文も用意して動作確認をしてください。

1. 長方形の幅と高さを入力して面積を戻り値として返す関数
   double rectangle(double w,double h)
2. 値 a,b を入力してその最小公倍数を戻り値として返す関数
   int getLCM(int a,int b)
3. 値 a,b,c を入力してそれらを大きい順に出力する関数
   void print_ascending_order(int a,int b,int c)

発展

- static 指定子
  関数内で定義した変数は関数が終了すると同時に破棄されますが、static 指定子を変数の定義の前に置くと関数が終了しても破棄されず値が保存されます。
  a06_5.c

```
#include <stdio.h>
void add(void);
```

（次ページ）

```
int main(void){
   int i;
   for(i = 0;i < 5;i++){add();}
   return 0;
}
void add(void){
   int a = 0;
   static int b = 0;                //最初の呼び出しの時のみ初期化
   a++;b++;
   printf("a = %d,b = %d¥n",a,b);
}
```

これをコンパイルして実行すると何となくつかめると思います。

- 再起呼び出し

関数はその関数の中でその関数自体を呼び出すことができます。これを再起呼び出しといいます。いかに例を示します。

a06_6.c

```
#include <stdio.h>
void print_number(int number,int n){
    if(number <= 0 && n <= 0){return;}
    else{print_number(number / 10,n - 1);}

    if(number > 0){printf("%d",number % 10);}
    else{printf("*");}
}
int main(void){
    int number;
    int n;
    scanf("%d%d",&number,&n);
    if(number > 0 && n > 0){print_number(number,n);}
    else{printf("error.¥n");}
return 0;
}
```

これはある数値と桁を入力してその数値の桁が入力した桁より小さいとき*で補うプログラムです。(正の値のみ)。print_number は number か n が 0 以上の時その値を 10 分の 1（小数点切り下げ）を渡してまた print_number を呼び出し、呼び出した関数

が終わると受け取った値の下一桁を出力して終わります(numberが0以下なら*を出力)。number、nの両方が0以下の時はそのまま終了します。以上の動作の繰り返しで出力します。

再起呼び出しを使うと以上のようにソースをきれいに描くことができます。その反面メモリをよく消費する、無限ループに陥りやすいなどの欠点があるので注意が必要です。