第６回　関数

今回は関数についてです。今まではあらかじめ用意されていたライブラリ関数（printf など）
を使用しプログラムを製作していましたが、関数自体を製作することを学びます。

値を戻す関数

```
#include<stdio.h>

int wa(int x, int y){    /*関数の定義      () 内のｘ、ｙを仮引数という*/
        int z;          z=x+y;
        return(z);        /*戻り値の指定*/
}

int main(void){
      int x,y,ans;
        printf("x+y を計算します");
        printf("整数ｘを入力してください¥n x=");
        scanf("%d",&x);
        printf("整数ｙを入力してください¥n y=");
        scanf("%d",&y);
        ans=wa(x,y);       /*関数 wa の呼び出し&ans へ wa の結果を代入*/
                          /*ここでのｘ、ｙを実引数という*/
        printf("x+y=%d です¥n",ans);

      return 0;
}
```

main 関数の上にある int　wa() {～} が自分で定義した関数です。関数は main 関数の外で

```
戻り値の型　関数名（引数の型　引数名１,引数の型　引数名２,・・）{
          関数の処理
return(戻り値)
}
```

と書きます。引数とは関数内の処理に使うデータのことで、戻り値は関数が返す値のことです。
上記の例ではｘとｙが引数でｚが戻り値です。例の流れは ans=wa(x,y)で関数 wa を呼び出し、ｘとｙを引数として関数 wa に渡し、ｘ＋ｙを計算しｚに代入します。ｚが戻り値なので、ans にｚの値が代入されます。

関数の呼び出しかたは

```
関数名　（引数１,引数２、・・・）;
```

です。

return の () 内には数式を入れることもできます。例では return(z) を return(x+y) に書き換えることができ、その場合は z の宣言が不要になります。

値を戻さない関数

```
#include<stdio.h>

void f(void){
printf("ソフトゼミ A　関数¥n¥n");
}

int main(void){

        f();

return 0;
}
```

戻り値がない関数について説明します。
上記の関数はソフトゼミ A　関数と表示するだけの関数です。
void f(void) の最初の void は戻り値がない、といった感じの意味で、戻り値がない場合 return() は省略できます。二つ目の void は引数を受け取らないといった感じの意味です。
引数がない関数を呼び出す場合は
関数名();
と書き、()の中身を書きません。

関数を活用することによって main 関数内の可読性があがり、プログラムの大まかな流れがつかみやすくなります。

関数のプロトタイプ宣言

上記の例ではプログラムを自作関数→main 関数の順序で書きましたが main 関数→自作関数だと
コンパイル時にエラーが出るはずです。これは main 関数→自作関数で書かれているとコンパイラは main 関数を先に読んで実行します。しかしこのときコンパイラは自作関数の存在をしらないので「こんな関数知らん」とエラー出しちゃうのです。これを防ぐためにプロトタイプ宣言を使います。プロトタイプ宣言はコンパイラに「後でこんな感じの自作関数を使うよ」と教えるような仕事をします。宣言の仕方は main、自作関数より上で

戻り値の型　関数名（引数の型　引数名１,引数の型　引数名２,・・）；
と書きます。関数の{処理}の部分を無くしただけのものです。
上記二つの例のプロトタイプ宣言は

```
int wa(int x, int y);          //値を戻す関数の例

void f(void);                  //値を戻さない関数の例
```

となります。

グローバル変数

```
#include<stdio.h>
int x;  /*グローバル変数としてｘの宣言*/

void f(void){
        x=10;
}
int main(void){
        x=1;
        printf("x=%d¥n",x);      /*x=1 が表示される*/
        f();                     /*x=10 が代入される*/
        printf("x=%d¥n",x);      /*x=10 が表示される*/
return 0;
}
```

グローバル変数とは main を含む関数の外で宣言された変数のことを言います。別名外部変数
グローバル変数はどこからでも変数をいじることができます。上記の例ではまずグローバル変数
ｘを宣言し、main 文内でｘに１を代入しそれを表示します。その後、関数ｆでｘに１０を代入し表示します。

結果、画面には
x=1
x=10
と出ます。

関数内で宣言された変数は局所変数と呼ばれます。局所変数は宣言された関数内でのみ使えます。そのため複数の関数で同じ変数名の変数を宣言することができ、それらは互いに影響を及ぼすことはありません。また、グローバル変数と局所変数でも同じ変数名を使うことができます。その場合はグローバル変数より局所変数が優先されます。以下例

```
#include<stdio.h>
int x;  /*グローバル変数としてｘの宣言*/

void f(void){
        x=10;
}
int main(void){
      int x;                ←ここに局所変数としてｘの宣言を追加した
        x=1;
        printf("x=%d¥n",x);
        f();
        printf("x=%d¥n",x);
return 0;
}
```

この結果は
ｘ＝１
ｘ＝１
となります。ｘ＝１で局所変数としてのｘに１が入りそれを表示。次に関数ｆでｘに１０を代入していますがここでのｘはグローバル変数のｘなので局所変数には影響ありません。次のprintfでは優先度が局所変数＞グローバル変数なので局所変数のｘが優先されｘ＝１が表示される。
という感じの流れです。

演習問題

x^2+y^2を計算する関数をつくり、ｘとｙをキーボードから入力し、計算結果を表示するプログラムを作ってみよう。