第６回　関数

プログラミングする上で効率よく作っていく為に、関数の考えかたはとても便利で有効的です。Ｃ言語には、標準で用意された素晴らしい関数(ライブラリ関数)がたくさんあるため、プログラミングが効率よくできるようになっています。関数は、ユーザが自ら新たな関数を作ることが出来ますので、見易さや機能性においても、有利にプログラミングが出来ます。

値を返す関数

```
#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=”);
        scanf(“%d”,&x);
        printf(“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。

関数名「wa」の左にある「int」が出力としての戻り値の定数 z 、整数型を表し、関数名「wa」の右のカッコ内が入力としての引数 x と y です。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。例の流れは ans= wa(x,y)で関数 wa を呼び

だし、 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 の後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ます。先ほどの関数の場合は z の宣言が不要になります。

値を戻さない関数

関数の入力に用いる変数を「引数」と呼び、関数の出力に用いる変数を「戻り値」と呼びます。int や double 型を使わない（変数がない）場合は、引数、戻り値ともに、「void」という言葉を用いて定義します。

```
#include<stdio.h>
void output(void){
        printf("ソフトゼミ A 関数¥n¥n");
}
int main(void){
        output( );
        return 0;
}
```

この関数は、『ソフトゼミＡ　関数』という表示を行うだけあって、特に戻り値を返す必要がありません。void とは、『空(から)』という意味です。void output(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 ( )は省略します。二つ目の void は引数を受け取らないという意味になります。

引数の使わ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「こんな関数知らないよ」とエラーが出てしまいます。
Ｃコンパイラは、前から順番に読み込んで翻訳していきます。しかし、Ｃコンパイラは自作関数の存在を知りません。自作関数の実体は main 関数のあとにあるからです。

そこで、main 関数、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。すると、Ｃコンパイラは後々になれば、自作関数があると信じてくれるのです。
これを関数のプロトタイプ宣言と言います。
例　　int wa(int x,int y);　　//値を戻す関数の例
　　void output(void);　　//値を戻さない関数の例

ヘッダとインクルード

プロトタイプ宣言によって、関数の引数や戻り値に関する仕様が与えられると、その関数を安心して呼び出せることがわかりましたね。

ライブラリ関数である printf や scanf などのプロトタイプ宣言は<stdio.h>の中で宣言せれています。そのための「おまじない」である、

　　#include<stdio.h>

という命令が必要なのです。

ライブラリ関数のプロトタイプ宣言などを含む<stdio.h>は、ヘッダとよばれ、それを #include 命令によって取り込むことをインクルードするといいます。

Ｃ言語では、計算技術などをサポートするために、平方根を求める sqrt 関数など、基本的な数学関数が用意されており、それらは<math.h>ヘッダ内で宣言されています。

```
#include<stdio.h>
#include<math.h>
double wa(int x);　　/*  プロトタイプ宣言　*/
int main(void){
　　int x;
　　double y;
　　printf("x=");
　　scanf("%d",&x);
　　y=wa(x);
　　printf("x の平方根は%f です。",y);
　　return 0;
}
double wa(int x){
　　double y;
　　y=sqrt(x);　　/* y へ x の平方根を代入*/
　　return y;
}
```

グローバル変数

グローバル変数は、メイン関数の外で定義することにより、どの関数からでも呼び出しができ、代入が可能で言葉どおり、「グローバル」な、広い範囲で変数が有効になることを意味しています。

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

上記の例ではまずグローバル変数 x を宣言し、main 文の内で 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;
}
```

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

$x^2 + y^2$ と $\sqrt{x^2 + y^2}$ を計算する関数を作り、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ろう。