第2回　GESセミナー　「関数」
～まとめと簡単な解説～

◆まとめ

・関数とは、小さなプログラムの塊。
・関数は、情報取得、処理、結果を返す、の流れ。
・使う前にプロトタイプ宣言が必要。
・書式は　返す結果の型　関数名(取得してくる情報の型　変数名);
・複数の関数で同じ変数を使う場合は、グローバル変数にする。

利点としては
・main()を短く出来る。
・同じ処理を書かなくてよくなる。
・プログラムを複数のファイルに分けることが出来るようになる。
等があります。

以下、簡単な解説です。

◆関数とは

今までは、main()の中に全てのプログラムを記述していたと思います。
小さなプログラムならそれで良いのですが、実際にゲーム等を作ると、
main()の中身が何千行にもなってしまい、とても大変です。
また、バグも出やすくなります。

そこで、まとまった処理を、ひとつの塊にしてやります。
そのプログラムの塊を、関数と言います。

プログラムをまとめたり、情報を渡して計算したり、その結果を返したり出来ます。

printfやrandも関数です。
渡した情報を表示するという機能をまとめたものがprintf関数で、
ランダムな値を計算して、結果として返すのがrand関数、という感じです。
実はmain()も関数です。

◆関数の流れ

関数の流れは大きく３つです。

①情報を取得する（関数に情報を渡す）

関数使用時に渡された情報を、変数(ローカル変数)として扱います。

これを「引数」と言います。複数指定することも可能です。

②処理をする

必要な計算などの処理をします。

③結果を返す。(関数を使った場所へ戻る)

計算などの処理をして出た答え等の結果を

関数を使った（呼び出した）場所へ返すことが出来ます。

これを「戻り値」と言います。ひとつしか返せません。

①と③は省略することも出来ます。

rand()のように処理には特に情報が必要ない時には①はいりませんし、

printf()のように計算などの結果が無い場合は③はいりません。

◆関数の作り方

①プロトタイプ宣言をする

関数を作るには、最初にどんな関数を作るのか宣言してやる必要があります。

「このプログラムではこんな関数作って使うよー」という宣言です。

これはmain()より上に記述します。

②関数の中身を作る

①で宣言したので、実際に作らないといけません。

関数での処理を書いていきます。

①の「宣言」に対して、「定義」と呼びます。

③複数の関数で使用する変数はグローバル変数にする

main()で作った変数を、作った関数の中で使いたい時、そのままでは使えません。

{}の中で作った変数の有効範囲は{}の中だけなので、

複数の関数で使う場合は、変数宣言を{}の外に出してやります。

こうすることで、どこからでも使える変数になります。

◆プロトタイプ宣言

書式は

返す結果の型　関数名(取得してくる情報の型　変数名);

です。

例としてゲームで、攻撃する処理まとめた attack という関数を作る場合、

```
void attack(void);
```

というふうになります。

void というのは、何もないということを表す型です。

つまり、返す結果と取得してくる情報は無い、という意味です。

もし、HP(hp)と攻撃力(at)が情報として必要な場合、

```
void attack(int hp, int at);
```

となります。

この様に、コンマで区切ることで、複数の情報を関数に渡すように指定出来ます。

さらに、処理した後の HP を int 型で結果として返したい場合、

```
int attack(int hp, int at);
```

となります。

つまり・・・

「int 型の hp と int 型の at という情報を取得して処理し、
結果を int 型で返す、　attack という名前の関数を作るよー」

と宣言しています。

◆関数の内容を書く

プロトタイプ宣言が出来たら内容を作ります。
今回の攻撃処理はHPから攻撃力を引くことにします。
これはmain(){}の下に書いてかまいません。

```
int attack(int hp, int at)
{
   int ans;             //計算結果の値を入れる変数
   ans = hp - at;       //渡されてきた情報の変数を使って、処理を行う
   return ans;          //結果の値を返す
}
```

return で結果を返すことが出来ます。
関数に渡す情報（引数）はコンマで区切ればいくつでも指定出来ますが、
返せる結果は１つだけです。

◆関数を使う（呼び出す）

関数を使うには、関数名を記述し、()の中に指定された情報を渡します。
printf等と同じような感じです。
「関数呼び出し」や「関数コール」と言ったりします。

```
main()
{
   int hp = 100;

   //関数呼び出し
   hp = attack(hp, 10); //HPと攻撃力を情報として渡し、帰ってきた結果をHPに代入

   //HP表示
   printf(“%d”, hp);  //90が表示される
}
```

◆使用例（情報渡して、結果を返す場合）

これまでのことを踏まえたプログラムの全体を見てみましょう。

```
//プロトタイプ宣言
//攻撃関数を作るよーと宣言してる
int attack(int hp, int at);

//メイン関数
main()
{
   int hp = 100; //グローバル変数ではない

   //攻撃関数呼び出し
   hp = attack(hp, 10);  //hp と攻撃力を情報として渡し、結果を hp に代入

   //HP を表示
   printf(“%d”, hp);
}

//宣言した通り攻撃関数を作る
int attack(int hp, int at)
{
   int ans;       //計算結果を入れる変数
   ans = hp - at; //攻撃処理
   return ans;    //呼び出した場所へ結果を返す
}
```

◆使用例（グローバル変数を使う場合）

変数をグローバル変数にすることで、その変数をどの関数からでも使えるようになって、
情報を渡したり、結果を返さなくても良くなり、簡単です。
main()等の関数の中括弧の外で変数を作ることでグローバル変数になります。

```
//プロトタイプ宣言
void attack(void);  //攻撃関数つくるよー宣言

//グローバル変数
int hp = 100;

main()
{
   //攻撃関数呼び出し
   attack();

   //HP 表示
   printf( “%d”, hp);
}

//宣言通り攻撃関数作ります
void attack(void)
{
   hp = hp - 10; //攻撃処理
}
```

グローバル変数はどこからでも使えるので便利ですが、逆にどこからでも使えてしまう為、
チーム制作などでは、他の人が勝手に使えてしまったりして、バグの原因になります。
なので、使わなくて済むなら使わない方が良いです。

◆グローバル変数についての補足

変数にはグローバル変数とローカル変数があります。
関数の中括弧の中で作られている変数がローカル変数、括弧の外で作られている変数がグローバル変数です。

```
int a;  //グローバル変数
main()
{
   int b; //ローカル変数
}
```

ローカル変数は、使えるのはその中括弧の中だけですが、
グローバル変数は、どこでも使えます。

main()と test()という関数があったとすると・・・

```
void test(void);  //プロトタイプ宣言

int a;  //グローバル変数

main()
{
   int b;       //ローカル変数(test 関数の内だけで使える)
   a = 10;      //OK!
   b = 10;      //OK!
   c = 10;      //エラー!
}

void test(void)
{
   int c;       //ローカル変数(test 関数の内だけで使える)
   a = 20;      //OK!
   b = 20;      //エラー!
   c = 20;      //OK!
}
```

関数に渡された情報(引数)は、ローカル変数として扱われます。

```
//プロトタイプ宣言
void test(int b);
void test2(int a);

//メイン関数
main()
{
   int a = 100;
   int b = 50;
   test(b);   //b を情報として渡す
   test2(a);  //a を情報として渡す
   printf(“%d”, a);  //100 が表示される
}

//test 関数本体(定義)
void test(int a)
{
   //ここで表示している a は、渡されてきた情報を
   //int 型の変数 a として取得したものなので、 main の a とは別のもの。
   printf(“%d”, a);   //100 ではなく、50 が表示される
}

//test2 関数本体(定義)
void test2(int a)
{
    //この a も引数(ローカル変数)の a で、main の a とは別のものなので、
    a = 0;       //ここで a を変更しても main の a には影響しない
}
```

以上です。
分からないところ、いまいち理解出来ないこと、その他質問等ありましたら、
放課後　北野館３階ゲームラボまで　気軽に来て下さい。

企画：GES