ソフトゼミ A 第 6 回

関数

プログラムは関数の組み合わせでできています。今までのゼミ A でも printf や scanf など様々な関数を使ってきましたが、なんと関数は自分で作ることもできるのです！！今日は自作関数を中心に扱っていきます。ゲーム制作でも自作関数は避けては通れないので頑張りましょう。

◆ そもそもまず、関数とは

基本的には『受け取った値に関数によって定められた操作をして、その結果の値を返す』というものです。関数が受け取った値を「引数」、関数が呼び出し元に返す値を「返り値」といいます。例えば、printf では「¥n や%d を含んだ出力したい文字列」「出力したい変数」が引数、「処理をした結果実際に出力された文字列」が返り値になります。一部には引数、返り値のどちらか、または両方が存在しない関数も存在します。

◆ main 関数とライブラリ関数

main 関数とは、「int main(void){}」で囲まれたひとくくりの部分のことです。その名の通りプログラムのメインで、プログラムは main 関数から実行されます。また、printf や scanf などはライブラリ関数と呼ばれ、printf や scanf は stdio.h（標準ライブラリ）を include することで使えるようになります。ライブラリには数学に特化した数学ライブラリ(math.h)やゲーム制作に向いた DX ライブラリがあります。特に、エレ研は DX ライブラリがゲーム制作の中心になります。

◆ 関数の定義、使い方

関数は main 関数の前で定義して使います。定義されなければ使えないので、関数内で別の関数を呼び出す場合は定義の順番にも注意が必要です(後述のプロトタイプ宣言を使う場合はこの限りではない)。実際に関数を利用したプログラムを見てみましょう。ここでは二通りの呼び出し方法を用いています。

```
/*Source.cpp*/
#include<stdio.h> //printf はここでインクルードして初めて使えるようになる。
//ここで関数 average を定義、3 つの整数の平均値を返す double 型の関数です。
double average(int a,int b,int c){ //ここで仮引数を宣言、ここでは int 型の 3 つ
	double d; //main 関数とあまり変わらない感覚でコーディングできます
	d = a + b + c;
	//関数内で有効な変数も宣言できるし、引数も変数と同様に使用できます。
	d /= 3;
	return d; //ここで戻り値を呼び出し元に返します。

};
int main(void) {
	int l, m, n, x, y, z;
	double o;
	l = 1;
	m = 4;
	n = 8;
	o = average(l, m, n); //ここで関数 average を呼び出している
	printf("l = %d, m = %d, n = %3.3f　平均は%d¥n", l, m, n, o);

	x = 11;
	y = 13;
	z = 7;
	printf("x = %d, y = %d, z = %3.3f　平均は%d¥n", x, y, z, average(x,y,z));
	//関数の中で関数を呼び出すこともできます
	return 0;
}
```

実行結果

```
l = 1, m = 4, n = 8　平均は 4.333
x = 11, y = 13, z = 7　平均は 10.333
続行するには何かキーを押してください . . .
```

このサンプルでは、3 つの整数の平均値を求めて double 形で返す関数 average を定義し、int 型の l、m、n の平均値を求めて double 形の o に代入しています。また、x、y、z についてのように printf 関数の中で average 関数を直接呼び出す、といった使い方もできます。

関数の定義と呼び出しの詳しい説明は以下の通りです。

```
//関数の定義
返り値の型 関数名(引数1の型 引数1の名前, 引数2の型 引数2の名前, …){
//ここでの引数は仮のもので、仮引数という。
        (関数の処理内容)
        return  返り値; //返り値がない場合は書かない
}

//関数の使用
変数名 = 関数名(引数1, 引数2,…);
//返り値を代入する変数と定義した返り値間、引数の型と仮引数間の型はそれぞれ一致することが望ましい。

変数名 = (関数名(引数1, 引数2,…) + 10)/3;
//上記の拡張版。ややこしいに見えるが、関数の使用と四則演算を組み合わせることもできるということである。

返り値の型 返り値名 =関数名(引数1, 引数2,…);
//こちらは変数の定義と同時に関数の戻り値を代入している。

printf("%d", 関数名(引数1, 引数2,…));
//printf関数の引数として関数を使用している。ここではint型だが当然他の型でも可能。
```

◆ 引数や返り値が存在しない場合

関数の定義の際に仮引数、返り値のどちらか、または両方が存在しない場合、以下のように「void」を用いて書きます。

```
返り値の型 関数名(void){ (関数の内容) return 返り値; }
//引数が存在しない場合

void 関数名(引数 1 の型 引数 1 の名前, 引数 2 の型 引数 2 の名前, …){ (関数の内容) }
//返り値が存在しない場合

void 関数名(void){ (関数の内容) }
//両方が存在しない場合
```

```
関数名(引数 1, 引数 2, …);
//返り値が void の場合の関数の呼び出し。返すべき変数が存在しないので、printf などのように処理として呼び出す。返り値に加えて引数が定義されない場合も同様。
```

返り値や引数が存在しない変数は、ポインタとの併用や、ゲームでの数値が絡まない処理に使われることも多いです。

◆ グローバル変数とローカル変数

関数内で宣言された変数をローカル変数といいます。初期値は不定なので、代入する必要があります。ローカル変数は定義されたその関数でのみ使え、一度呼び出しを終えた後に再び呼び出しても基本的には値は保持されません(static int など例外あり)。ポインタを用いない限りは基本的に外部の関数から書き換えることはできず、別の関数間では同じ名前のローカル変数を使うことができます。

関数の外で定義されプログラム全体で共有、関数をまたいで使える変数をグローバル変数といいます。初期値は 0 です。グローバル変数はどこからでも参照や変更ができますが、どこで値が変わったのかわかりづらくなってしまいます。バグの修正(デバッグ)に支障をきたすので、多用するのはなるべく避けましょう。

また、グローバル変数と同じ名前のローカル変数が関数内にある場合にはローカル変数が優先されます。

◆ 練習問題

以下の関数を作成し、1 つのプログラム(main 文)でそのすべてを呼び出しなさい。どの関数を呼び出したのかわかりやすくすること。

1. double 型で円の直径を入力し、その面積を求める関数。円周率は 3.14 とする。
2. int 型の変数 3 つを引数とし、その最小値を返し値とする関数。
3. int 型の自然数ひとつを引数とし、その階乗を返し値とする関数

ソフトゼミＡ第６回

問題の解答

◆ 練習問題

以下の関数を作成し、1 つのプログラム(main 文)でそのすべてを呼び出しなさい。どの関数を呼び出したのかわかりやすくすること。

1. double 型で円の直径を入力し、その面積を求める関数。円周率は 3.14 とする。
2. int 型の変数 3 つを引数とし、その最小値を返し値とする関数。
3. int 型の自然数ひとつを引数とし、その階乗を返し値とする関数

※回答の一例です。正しい動作さえすれば関数の呼び出し方法などは問いません。

```
#include<stdio.h>
double circle(double d) {
        d /= 2;
        return d*d*3.14;
}
int min(int a, int b, int c) {
        int smallest = a;
        if (b < smallest)smallest = b;
        if (c < smallest)smallest = c;
        return smallest;
}
int kaijyo(int a) {
        int i, b = 1;
        for (i = 1; i <= a; i++)b *= i;
        return b;
}
//次のページに続く
```

```
//回答続き
int main(void) {
        double a;
        int b, c, d, e, f;
        printf("円の直径a(double型)を入力：");
        scanf("%lf", &a);
        printf("直径 %3.3f の円の面積は %3.3f ¥n",a,circle(a));

        printf("int型の整数b,c,dを入力：");
        scanf("%d%d%d", &b, &c, &d);
        printf("%d, %d, %dのうち最少は %d ¥n", b,c,d, min(b,c,d));

        printf("int型の整数eを入力：");
        scanf("%d", &e);
        f = kaijyo(e);
        printf("f = e! = %d! = %d ¥n", e, f);

        return 0;
}
```

ソフトゼミ∀第６回

関数

◆ 配列引数について

```
返り値の型 関数名(引数の型 引数の名前[],…) { (関数の内容) return 返り値; }
//関数を定義するとき。返り値の型として void を使用可。

代入先 = 関数名(引数名,…)
//関数を呼び出すとき。[]を付けないことに注意。変数の宣言と同時の代入、四則演算との併用、型さえ合えば別の関数の引数として呼び出すことも当然可能
```

配列を引数とする場合は以下のように関数の定義や呼び出しをします。

配列引数を使用した場合、関数で使用した配列を書き換えると呼び出した元の配列も書き換わってしまいます。これは注意すべき点にもなりますが、逆手にとって仮引数に配列を複数定義して、処理結果を書き込みたい配列を引数に指定すれば、処理した結果を配列として書き出し、疑似的な返り値とすることもできます。

元の配列が書き換わるのを避ける場合は、const という型修飾子を用いて以下のように記述することで、今までの変数と同じように使えます。

```
返り値の型 関数名(const 引数の型 引数の名前[],…) { (関数の内容) return 返り値; }
```

また、多次元配列の場合は関数の定義の際、以下のように先頭以外の要素を指定する必要があります。

```
返り値の型 関数名(引数の型 引数の名前[][要素],…) { (関数の内容) return 返り値; }
```

◆ プロトタイプ宣言

このように関数の宣言だけを先に行い、main 関数の後に実際の処理内容を書くことをプロトタイプ宣言といいます。

```
返り値の型 関数名1(引数1の型 引数1の名前, 引数2の型 引数2の名前, …);
//関数名1のプロトタイプ宣言
返り値の型 関数名2(引数aの型 引数aの名前, 引数bの型 引数bの名前, …);
//関数名2のプロトタイプ宣言
int main(void) {
        (main 関数の処理内容)
        return 0;
}
返り値の型 関数名1(引数1の型 引数1の名前, 引数2の型 引数2の名前, …){
//こちらにもプロトタイプ宣言時と同じ引数を記述する必要がある。
        (関数の処理内容)
        return　返り値;
}
返り値の型 関数名2 (引数aの型 引数aの名前, 引数bの型 引数bの名前, …){
        (関数の処理内容)
        return　返り値;
}
```

二度手間にはなりますが、プログラムが見やすくなるメリットがあります。また、プロトタイプ宣言を用いることにより関数を宣言する順番に気を使う必要もなくなります