第４回C言語講座

１．　文字列

文字列とは、「文字の列」です。これはどういうことかというと、文字列とは「文字の配列」ということです。例えば、”abc”は a b c ¥0（¥0 は終了コード）の4つの char 型の配列で出来ています。文字列はメモリ上に文字単位で連続的に格納されているのです。さらに、文字は文字コードとして格納されるので、この場合は 97 98 99 0 と保存されます。

文字列が文字の配列であるということを以下のサンプルに示します。

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*
C言語講座サンプル1
文字列
by y.t
*/
#include <stdio.h> /*入出力のヘッダーファイル*/
/*メイン関数*/
int main()
{
	char str[7] = "sofume";	//s o f u m e ¥0
	int index;
	printf("%s¥n", str);
	printf("取り出したい文字は何文字目？¥n");
	scanf("%d", &index);
	printf("%s の%d 文字目は%c¥n", str, index, str[index]);
	return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

配列は 0 から始まることに注意してください。0 と入力すると s が、3 と入力すると u が取り出されます。

文字列の表現方法はポインタで詳しくやります。

２．　関数

今までは printf 関数や scanf 関数などあらかじめ用意されていた関数を使いました。今回は自分で関数を作ります。頻繁に使う処理は関数にしておくと、そのつど処理の実装を書く必要が無く、関数を呼び出すだけで済みます。

関数を作るには作りたい関数の宣言をする必要があります。変数も宣言しないと使えませんよね。それと同じです。関数を宣言したら関数の定義、つまり中身を書きます。これで、作った関数が使えるようになります。では、サンプルを見てみましょう。

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

2006 第4回C言語講座（By y.t）

```
/*
C言語講座サンプル2
関数
by y.t
*/
#include <stdio.h> /*入出力のヘッダーファイル*/
/*メイン関数*/
int add(int a1, int a2);      //関数の宣言
float ave(float sum, float num);      //関数の宣言
int main()
{
        int a1, a2;
        int s;
        float a;
        printf("2つの数字を入力してください¥n");
        scanf("%d %d", &a1, &a2);
        s = add(a1, a2);
        a = ave(s, 2);
        printf("合計は%d¥n", s);
        printf("平均は%.2f¥n", a);
        return 0;
}
//add 関数の定義
int add(int a1, int a2)
{
        return a1 + a2;
}
//ave 関数の定義
float ave(float sum, float num)
{
        return sum / num;
}
```

//

関数の宣言は main 関数の前で行います。そして、関数の定義は関数の宣言後に書きます。

関数は

返す値の型　関数名　（引数の型　引数...）

という書式で成り立っています。

関数が値を返すときには return を使い、返す値を記述します。

３．　構造体

構造体とは個々の変数を大きな枠でくくったものです。と言ってもなにやら難しそうですね。その辺はサンプルを見てもらいましょう。

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*
C言語講座サンプル3
構造体
by y.t
*/
#include <stdio.h> /*入出力のヘッダーファイル*/
#include <stdlib.h>
#include <string.h>
//構造体の宣言
struct HERO
{
	int hp;
	int mp;
	int atk;
	int def;
};
/*メイン関数*/
int main()
{
	struct HERO hero;
	char name[64];
	int nameSize;
	printf("名前を入力してください¥n");
	scanf("%s", name);
	nameSize = strlen(name);//strlen：文字列の長さを取得する
	srand(nameSize);
	hero.hp = rand() % name[0];
	hero.mp = rand() % name[nameSize - 1];
	hero.atk = rand() % name[nameSize / 2];
	hero.def = rand() % name[nameSize / 3];
	printf("HP = %3d¥nMP = %3d¥nATK = %3d¥nDEF = %3d¥n", hero.hp, hero.mp, hero.atk,
hero.def);
	return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

RPGを例にとりましたが、HPやMPといったデータはプレイヤーにも敵にも存在します。それなのにhero_hpや enemy_hp というような変数を宣言しているとプログラムが大きくなってきたときに混乱が生じることがあります。hero や enemy を構造体にしてその中に必要な変数を宣言するとデータの構造がはっきりします。プログラミングでは上手なデータ構造の設計が出来るとプログラムが読みやすくなったり、プログラムが書きやすくなったりします。

構造体に含まれている変数をメンバ変数といいます。また、struct に続く文字列をタグ名といいます。

サンプルのような構造体の宣言の仕方はあまりしません。typedef というものを使って構造体をデータ型として定義します。以下はそのサンプルです。

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/*
C言語講座サンプル4
構造体2
by y.t
*/
#include <stdio.h> /*入出力のヘッダーファイル*/
#include <stdlib.h>
#include <string.h>
//構造体の宣言
typedef struct HERO
{
	int hp;
	int mp;
	int atk;
	int def;
}HERO; //struct HERO は HERO というデータ型として扱う。
/*メイン関数*/
int main()
{
	HERO hero;	//HERO 型の構造体を宣言
	char name[64];
	int nameSize;
	printf("名前を入力してください¥n");
	scanf("%s", name);
	nameSize = strlen(name);//strlen：文字列の長さを取得する
	srand(nameSize);
	hero.hp = rand() % name[0];
	hero.mp = rand() % name[nameSize - 1];
	hero.atk = rand() % name[nameSize / 2];
	hero.def = rand() % name[nameSize / 3];
	printf("HP = %3d¥nMP = %3d¥nATK = %3d¥nDEF = %3d¥n", hero.hp, hero.mp, hero.atk,
```

```
hero.def);
        return 0;
}
```

//

このように構造体をデータ型として定義すると、構造体を変数の感覚で使うことが出来ます。

４．　本日の演習

1. 入力された小文字を大文字に変換するプログラムを作ってみてください。（ヒント：小文字の文字コードから 32 を引くと・・・）

2. 関数 $F(x) = 3X^2 + 4$ を実装してください。

3. 学年、番号、名前をメンバに持つ構造体を作ってください。

５．　本日の関数

表1. 本日の関数

関数	引数	戻り値	処理内容
size_t strlen(const char *string);	string 終了コードで終了する文字列	string の文字数	string で渡された文字列の文字数を数えます。 必須ヘッダー string.h