C 言語 011 関数の作成

【関数の作成】

関数は自作できます。自作した関数を「ユーザ定義関数」と称します。

＜関数の定義＞

```
戻り値のデータ型 関数名(引数のデータ型 仮引数名) {
    処理を記述
    return 戻り値;
}
```

引数は複数設定できますが、戻り値はひとつしか返せません。

＜引数が無い関数の定義＞

```
戻り値のデータ型 関数名(void) {    /* 引数に void と指定 */
    処理を記述
    return 戻り値;
}
```

＜戻り値が無い関数の定義＞

```
void 関数名(引数のデータ型 仮引数名) {    /* 戻り値のデータ型に void と指定 */
    処理を記述
}
```

return は不要です。記述した処理を行うだけの関数になります。

引数も戻り値も両方無い関数は両方とも void を指定します。

＜仮引数と実引数＞

関数の定義で宣言する引数を「仮引数」、関数の使用に指定する引数を「実引数」と称します。

```
戻り値のデータ型 関数名(引数のデータ型 仮引数名) {     /* 関数の定義 */
変数 = 関数名(実引数);                                 /* 関数の使用 */
```


Copyright 2018 DoubleQuotation Limited.
20181225

＜例文１＞

三角形の面積を求める関数。

```
/* ユーザ定義関数 */
double triangle(double base, double height) {    /* 引数が複数の場合はカンマで区切る */
    double answer;
    answer = base * height / 2;                    /* 引数を演算して */
    return answer;                                 /* 戻り値として返す */
}

/* メイン関数 */
int main(void) {
    double area;                    /* 戻り値を入れる変数 */
    area = triangle(10, 7);         /* base に 10、height に 7 を指定 */
    printf("面積は%f¥n", area);     /* area には 35 が代入されている */
    return 0;
}
```

【プロトタイプ宣言】

「定義」は処理や実体を記述すること、「宣言」はコンパイラに対して使用することを表します。
関数 A で関数 B を使用したい場合、関数 B（使われる方）は関数 A（使う方）より先に定義しなければなりません。
例文 1 ではメイン関数が triangle 関数を使用しているので、triangle 関数はメイン関数よりも先に定義しています。
メイン関数よりも後に triangle 関数を定義した場合は「プロトタイプ宣言」が必要になります。

＜関数のプロトタイプ宣言＞

関数のプロトタイプ宣言は変数宣言と同様、宣言部で行うこと。

```
戻り値のデータ型 関数名(引数のデータ型 仮引数名…);
もしくは
戻り値のデータ型 関数名(引数のデータ型 …);     /* 仮引数名は省略できる */
```


Copyright 2018 DoubleQuotation Limited.
20181225

＜例文 2＞

```
/* メイン関数を先に定義 */
int main(void) {
    double triangle(double, double);    /* 関数のプロトタイプ宣言 */
    double area;
    area = triangle(10, 7);             /* 宣言していないと triangle は使用できない */
    printf("面積は%f¥n", area);
    return 0;
}

/* ユーザ定義関数を後に定義 */
double triangle(double base, double height) {
    double answer;
    answer = base * height / 2;
    return answer;
}
```

【関数の終了】

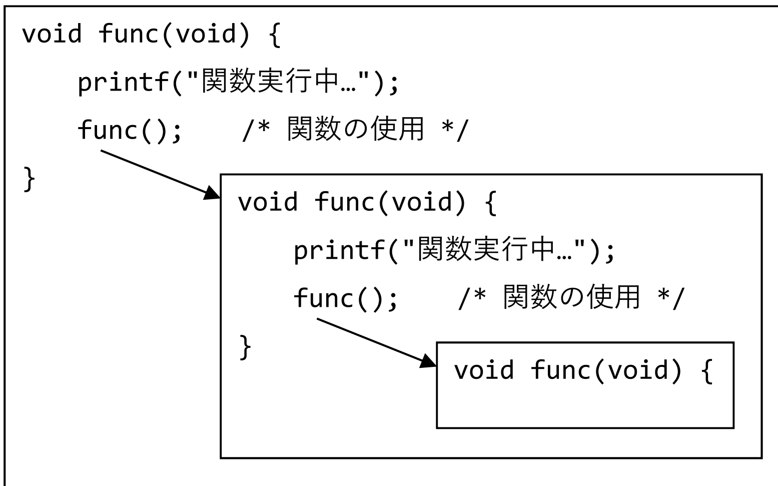
関数は「return 戻り値;」で関数内の処理を終了します。
戻り値が void でも「return」で関数を終了できます。

```
void func(int arg) {
    if(arg == 0) {
        return;    /* 変数 arg が 0 なら関数を終了 */
    }
（以下略）
```


Copyright 2018 DoubleQuotation Limited.
20181225

【再帰処理】

関数 A から関数 A を使用する。自関数内で自関数を使用することを「再帰処理」と称します。
再帰処理はさまざまなアルゴリズムで使用されますが注意が必要です。



関数は使用するたびにメモリを消費します。
上記のコードでは永久ループのごとく、ひたすら func 関数を実行し続けてしまうので、いずれメモリを使い果たすエラーが発生します。
この場合はループ文の break のように、条件により「return」で関数の処理を終えるような作りにしないといけません。

＜例文＞
再帰処理を利用した引数 n の階乗計算を行う factorial 関数

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    }
    else {
        /* 再帰処理を行っている */
        return factorial(n - 1) * n;
    }
}
```


Copyright 2018 DoubleQuotation Limited.

C 言語 011 関数の作成

【メイン関数】

「int main(void)」で始まる関数を「メイン関数」と称し、特別な役割を持っています。
メイン関数はどのプログラムでも必ず記述する必要があり、プログラムの最初に実行される関数です。

＜メイン関数の引数＞

```
int main(int argc, char* argv[]) {
```

メイン関数にも引数を設定できます。メイン関数の引数は「int 型」と「文字配列のポインタ型」の 2 つを宣言します。仮引数名は自由ですが一般的には上記のような「argc」「argv」が使用されます。

＜例文＞

```
#include <stdio.h>
int main(int argc, char* argv[]) {    /* メイン関数に引数を設定 */
    int i;
    for (i = 0; i < argc; i++) {
        printf("引数%d 番目：%s¥n", i, argv[i]);
    }
    return 0;
}
```

メイン関数の実引数の指定はコマンドから行います。
コマンドから実行ファイルを指定する時に「実行ファイル名 引数」と指定します。
「実行ファイル名 引数 引数」と各引数をスペースで空けて入力すれば何個でも指定できます。
変数「argc」は入力された実引数の数、配列「argv」には入力された値が文字列で記録されます。

＜実行結果＞

例文をコンパイルし「sample.exe」を作成。

```
C:¥sauce¥c>sample aaa 100 10.5
引数0番目：sample
引数1番目：aaa
引数2番目：100
引数3番目：10.5
```

コマンドから「aaa」「100」「10.5」の 3 つの値を引数にして実行します。
引数には実行ファイル名も含まれ、変数 argc は引数の数「4」、変数 argv[0]は実行ファイル名、argv[1]は「aaa」、argv[2]は「100」、argv[3]は「10.5」が代入されています。
argv は文字配列なので数値で使用したい場合は変換する必要があります。

Copyright 2018 DoubleQuotation Limited.
20181225

＜メイン関数の戻り値＞

メイン関数にも戻り値を設定できます。戻り値を設定した場合は「return」で値を返し、設定しない場合は戻り値のデータ型を「void」にします。

今までのテキストでは「return 0」で整数 0 を返していました。一般的にはメイン関数の戻り値は int 型で正常終了なら 0、異常終了なら 0 以外を返します。

メイン関数の戻り値のデータ型を int 以外に設定した場合、MinGW では Warning（警告）が出ますが、エラーにはなりません。

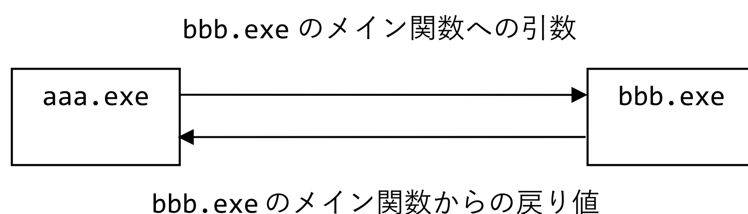
＜実行ファイルの呼び出し＞

```
#include <stdio.h>
#include <stdlib.h>
void main(void) {
    int ret = system("*** 引数");    /* system は引数に指定したコマンドを実行する関数 */
    printf("実行結果:%d¥n", ret);
}
```

＜system 関数＞

定　　義	int system(const char* str);
ﾍｯﾀﾞｰﾌｧｲﾙ	stdlib.h
説　　明	引数に指定したコマンドを実行する関数。コマンドなので実行ファイル名でプログラムを実行することができ、「実行ファイルのメイン関数」へ引数を指定できる。
戻　り　値	実行した「実行ファイルのメイン関数」の戻り値。

メイン関数の引数、戻り値を利用すれば実行ファイル同士で値の受け渡しができます。



Copyright 2018 DoubleQuotation Limited.
20181225