関数 2

前回は関数の基本的な使い方について習った。今回は具体的な利用例について習っていく。

プロトタイプ宣言の引数の変数名の省略

次に示す例は前回のテキストのプログラム例２について少し変更を加えたものである。

プログラム例１

```
#include <stdio.h>

int add(int, int); // プロトタイプ宣言

int main(void) // メイン関数
{
        int a = 3, b = 4, c;
        c = add(a, b);
        printf("a+b=%d¥n", c);
        return 0;
}

int add(int x, int y) // 関数本体の定義
{
        int z;
        z = x + y;
        return z;
}
```

実行結果

```
a+b=7
```

着目するべきところはプロトタイプ宣言である。プロトタイプ宣言では引数の型と数をコンパイラがチェックすることになるため、引数の変数名はあまり意味が無い。そのため、引数の変数名は省略することができる。

```
前回までの使い方        int add(int x, int y);
今回の使い方            int add(int, int);
```

もちろん、関数本体int add(int x, int y){･･･}では引数の変数名を用いてプログラムを書いていくことになるので、省略することはできない。

関数名の重複の禁止

プログラム例 1 では int 型の返却値と引数を持ち、2 つの引数の合計を求めるプログラムについて示した。ここでは double 型の返却値と引数を持ち、同様の処理を行う関数を追加する。この場合、double add(double x, double y);のように同じ関数名を用いることはできず、プログラム例２に示すように関数名を別のものにしなくてはならない。

プログラム例２

```
#include <stdio.h>

int add_int(int, int);                  // プロトタイプ宣言
double add_double(double, double);      // プロトタイプ宣言

int main(void) // メイン関数
```

```
{
        int a = 3, b = 4, c;
        double d = 3, e = 4, f;

        c = add_int(a, b);
        printf("a+b=%d¥n", c);

        f = add_double(d, e);
        printf("d+e=%f¥n", f);

        return 0;
}

int add_int(int x, int y) // 関数本体の定義
{
        int z;
        z = x + y;
        return z;
}

double add_double(double x, double y) // 関数本体の定義
{
        double z;
        z = x + y;
        return z;
}
```

実行結果

```
a+b=7
d+e=7.000000
```

プログラム例２において、四角で囲んだ部分を確認する。add_int と add_double では引数や返却値に用いている変数名（x、y、z）を同じにしたがコンパイルは通る。変数には有効範囲というものが存在し、これを**通用範囲（スコープ）**と呼ぶ。それぞれの関数で宣言した変数はそれぞれの関数内で利用することができ、それぞれ独立して扱われる。

関数から関数を呼び出す

関数の内部でさらに関数を呼び出すこともできる。以下に簡単な例を示す。

プログラム例３

```
#include <stdio.h>

int function1(int);             // プロトタイプ宣言
int function2(int);             // プロトタイプ宣言

int main(void) // メイン関数
{
        int a = 3, b;

        b = function1(a);
        printf("%d¥n", b);

        return 0;
```

```
}

int function1(int x) // 関数本体の定義
{
        int y;
        y = function2(x);

        return y;
}

int function2(int s) // 関数本体の定義
{
        int t;
        t = s * s;

        return t;
}
```

実行結果

```
9
```

このプログラムでは以下のように実行されていく。

(1)main 関数で function1 関数が呼び出される。実引数は a であり、3 が格納されている。
(2)function1 の関数本体において、仮引数 x に 3 が代入されて function1 の関数の処理が実行される。
(3)実引数 x として function2 関数が呼び出される。
(4)function2 の関数本体において、仮引数 s に 3 が代入されて function2 の関数の処理が実行される。
(5)fucntion2 の関数本体において、変数 t が返却され、function1 の関数本体の変数 y に値が代入される。
(6)function1 の関数本体において、変数 y が返却され、main 関数において、変数 b に値が代入される。

演習

・プログラム例１～３を作成し、動作を確認しなさい。

・教科書 p.171 のリスト 5.9 のプログラムを作成し、出力を確認しなさい。このプログラムは整数の正のべき乗を計算する関数を用いたプログラムである。P.165 のリスト 5.3 にはプロトタイプ宣言を用いない場合のプログラムがある。こちらは動作についての解説が示されているので、良く読んで、プログラムの具体的な実行順を把握しておくこと。

・教科書 p.173 のリスト 5.10 プログラムを作成し、出力を確認しなさい。このプログラムは組合せを計算するプログラムである。「組合せ」とは例えば、4 文字 A,B,C,D があるとき、この中からいくつかの文字を選択したとすると、その組合せパターンはいくつあるかを求めるものである。選択した文字の並び順については区別しない（AB と BA は同じものとして扱かう）。組合せの式は以下で表される。

$$ {}_nC_r = \frac{n!}{r!(n-r)!} $$

例えば、選択する文字数を 2 とすると r=2、文字総数は n=4 となるので、

$$ {}_4C_2 = \frac{4!}{2!(4-2)!} = 6 $$

となって、6 通りの組合せとなる。
　また、教科書 p.185 のリスト 5.16 のプログラムを作成し、出力を確認しなさい。このプログラムも組合せを求めるプログラムであるが、関数の中から関数を呼び出すことにより、簡潔なプログラムを実現している。

課題 1

　以下の課題では main 関数を含めるなど、動作が可能なプログラムにして提出すること。

Kadai1-1　3 つの int 型整数の最小値を返す関数を作成しなさい。
　プロトタイプは次のようになる。

```
int min3(int x, int y, int z);
```

　キーボードから値を３つ入力し、これらの関数に引数で値を渡す。関数で処理した結果を表示すること。

Kadai1-2　2 つの整数の和、差を求める関数を作成しなさい。
　プロトタイプは次のようになる。

```
int sum(int x, int y);
int diff(int x, int y);
```

　キーボードから値を２つ入力し、これらの関数に引数で値を渡す。関数で処理した結果を表示すること。

「関数」は続きがあるので、締切は後で示す。