第 5 章　自作の関数(サブルーチン)を作成しよう

概要

プログラムを作成する際に同じ処理をさせる命令を何度も書くのは、煩雑であるばかりでなく、プログラムが長くなって処理の流れが分りにくくなります。
ここではこの問題を解決する関数の作成について学びます。

この章の目標

プログラムを複数の処理の塊(関数)に分ける方法を覚える。
関数の引数と返り値について理解する。
関数の間における変数の共有と値の保存について覚える。

5.1　処理の下請け、関数を作ろう

---

7 章　下請けを作ろう　ーサブルーチンと関数ー[pp. 85-99]

注意：参考書では、サブルーチンという用語を用いていますが、多くの他の参考書では、関数という用語で表現されています。従って、この演習では関数という用語を使用して話を進めていきます。

●関数とは？
第 4 章までに習った変数や制御構文を用いることによって､かなり込み入ったプログラムを書くことができます。しかし、プログラムが大きくなると同じ処理の繰り返しを何回も記述しなければならなかったり、瑣末的な処理まで記述しなければならないような場合が生じてきます。こうなると、プログラム全体を把握しにくく、プログラム開発が困難となります。
そこで、処理の塊毎にまとめて表現する方法が必要となりました。この処理の塊を C 言語においては、関数として表現します。
数学における関数とは意味が異なり、ここでの関数とは「ある一つの仕事を実行する自己完結した処理の集まり」をいいます。たとえていうなら､”必要な情報”を与えて「お願いします」というときちんと仕事をして”結果”を”返して”くれる、頼りになる部下といったところです。
この部下（関数）に与える必要な情報（値）のことを引数といい、関数が返してくれる値のことを戻り値（返り値)、仕事をお願いすることを関数呼び出しといいます。

●関数の定義
サブルーチン[p. 87]
引数[p. 90]、注意[p. 92]
返り値を渡すサブルーチン[pp. 94-95]

○引数、返り値がない場合
・書式

```
void　関数名(void)
{
        宣言と文
}
```


Hirayuki Hirokawa
http://192.244.150.43/

・例

```
void hello(void)
{
        printf("Hello!!¥n");
}
```

○引数、返り値がある場合

・書式

```
return 型　関数名(引数の宣言)
{
        宣言と文
        return 返り値;
}
```

・例

```
int plus(int a,  int b)
{
        int x;

        x = a + b;
        return x;
}
```

●プログラム例

(P 例 : 5.1-1)

```
#include<stdio.h>

int plus(int, int);

main()
{
        int a,b,c;

        scanf("%d%d", &a, &b);
        c = plus(a,b);
        printf("%d + %d = %d¥n", a, b, c);
}

int plus(int x, int y)
{
        int z;

        z = x + y;
        return(z);
}
```

注意 : この例のように、return の後に括弧() をつけることがよくありますが、括弧はなくてもかまいません。

●補足

注意[p. 89]

ステップアップ : サブルーチンを書く順番[p. 99]

Hirayuki kurokawa
http://192.244.150.43/

5.2　変数の管理者を決めよう

8 章　変数の持ち主を決めよう　ー大域変数と局所変数ー[pp.100-106]

●この節のポイント
関数が共有する変数と共有しない変数があることを理解する。
関数終了後も値が保存される変数があることを理解する。

●大域変数と局所変数
(P 例 : 5.2-1)

```
#include<stdio.h>

int x; /* 大域変数「x」 */

void sub1(void);

main()
{
        x = 10;
        printf("main:%d¥n",x);
        sub1();
}

void sub1(void)
{
        int y; /* sub1 関数の中の局所変数「y」 */

        y = x*x;
        printf("sub1:%d,%d¥n",x,y);
}
```

(P 例 : 5.2-2)

```
#include<stdio.h>

int x; /* 大域変数「x」 */

void sub2(int);

main()
{
        x = 10;
        sub2(x);
        printf("main:%d¥n",x);
}

void sub2(int x) /* sub2 関数の中の局所変数「x」 */
{
        x += 10;
```


Hirayuki hurokawa
http://192.244.150.43/

```
        printf("sub2:%d¥n",x);
}
```

●静的変数
値が保存される変数[p.104]

(P 例 : 5.2-3)

```
#include<stdio.h>

void plus(int);

main()
{
        int i;

        for(i=0;i<5;i++)
                plus(i);
}

void plus(int a)
{
        int b = 0;
        b = b + a;
        printf("%d:%d¥n",a,b);
}
```

(P 例 : 5.2-4)

```
#include<stdio.h>

void plus(int);

main()
{
        int i;

        for(i=0;i<5;i++)
                plus(i);
}

void plus(int a)
{
        static int b = 0; /* 静的変数 */
        b = b + a;
        printf("%d:%d¥n",a,b);
}
```

Hirayuki hurokawa
http://192.244.150.43/