関数の演算結果を返却する

C 言語では、引き数を介して関数にデータを受け渡す。関数内部の演算結果を関数呼び出し側に返すには二通りのやり方がある。

1) 関数の戻り値（返却値）として返す
2) 引数を介して返す

```
double sqr(double);

main()
{
  double x = 2.0, y = 0.0;

  y = sqr(x);
  printf("%f %f\n", x, y);
}

double sqr(double x)
{
  return x*x;
}
```

演算結果を関数の戻り値として返す例

実引数 x を関数 sqr に受け渡し、結果は関数の戻り値として y に代入される。

引数を介して演算結果を返す

関数の戻り値として演算結果を返す場合、戻り値の数は1つに限られる。また、複雑なデータ構造を戻り値とすることはややこしい（不可能ではない）

引数を介して演算結果を返す場合には、参照渡しを用いる（ポインタを使う）

```
void sqr_new(double, dobule*);

main()
{
  double x = 3.14156, y = 0.0;

  sqr_new(x, &y);

  printf("%f %f\n", x, y);
}

void sqr_new(double x, double *y)
{
  *y = x*x;
}
```

関数 sqr_new を 2 つの引数で呼び出す。第 2 引数に第 1 引数の値の 2 乗が格納される。

不良例

```
void sqr_new(double, dobule);

main()
{
  double x = 3.14156, y = 0.0;

  sqr_new(x, y);

  printf("%f %f\n", x, y);
}

void sqr_new(double x, double y)
{
  y = x*x;
}
```

正しく動かない例である

データは値渡しで関数に受け渡される

関数 sqr_new 内部のローカル変数 y の値は x*x になるが、関数呼び出し元の変数 y は不変。

値渡しでは、関数呼び出し側の引数の値は不変。

配列名 = ポインタ

配列名は、その配列へのポインタである。従って関数の引数として配列名を与えると、参照渡しになる。
（関数内部で配列要素の値を変更すると、関数呼び出し側に反映される）

```
void manipulate_vector(double[], int size);

main()
{
  double x[3] = {0.0, 0.0, 0.0};

  display_vector(x, 3);
  manipulate_vector(x, 3);
  display_vector(x, 3);
}

void manipulate_vector(double x[], int size)
{
  int i;
  for(i=0; i<size; i++)
    x[i] += 1.0;
}
```

Gauss の消去法

前進消去過程を行う関数 gauss に、連立 1 次方程式を定める行列 A、ベクトル b を与える。上三角化された連立 1 次方程式を表す行列 A_after、ベクトル b_after が引数を介して返却される。

前進消去過程で軸がゼロになり上三角化に失敗した場合を考慮して、
関数 gauss の戻り値は、int 0 （失敗）、int 1（成功）とする。

```
#define SIZE 3 /* 次元 n の指定 */
...

main()
{
  double a_after[SIZE][SIZE], b_after[SIZE], x[SIZE];

  report(a, b);
  if( gauss(a, b, a_after, b_after) != 0){
    report(a_after, b_after);
    solve(a_after, b_after, x);
  } else printf("解けませんでした！\n");
}
```

5