线性卷积概述

在信号处理中，**卷积**（Convolution）是两个信号之间的运算，通常用于滤波器的设计与信号处理。线性卷积的数学定义是：

$$y[n] = \sum_{k=0}^{M-1} x[k] \cdot h[n-k]$$

其中，`x` 是输入信号，`h` 是滤波器（或系统的响应），`y` 是输出信号。

在这段代码中，`processing3` 实现了卷积操作，其中 `input1` 和 `output2` 被用来进行卷积运算，结果存储在 `output4` 中。

`processing3` 子程序分析

```
static int processing3(int *input1, int *output2, int *output4)
{
    int m = sk;
    int y = zhy;
    int z, x, w, i, f, g;

    // 正序卷积计算部分
    for(;(m - y) > 0;)
    {
        i = y;
        x = 0;
        z = 0;
        f = y;
        // 内层循环：计算卷积的累加
        for(; i >= 0; i--) {
            g = input1[z] * output2[f]; // 计算输入信号和滤波器的乘积
            x = x + g;  // 累加到 x 中
            z++;  // 输入信号指针向后移动
            f--;  // 滤波器指针向前移动
        }
        *output4++ = x;  // 将结果存入输出缓冲区，并移动输出指针
        y++;  // 向后移动输出的指针
    }

    // 反向卷积计算部分（倒序）
    m = sk;
    y = sk - 1;
    w = m - zhy - 1;
    for(; m > 0; m--) {
        y--;
        i = y;
        z = sk - 1;
        x = 0;
        f = sk - y;
        // 反向循环：计算卷积的累加
        for(; i > 0; i--, z--, f++) {
            g = input1[z] * output2[f];  // 计算输入信号和滤波器的乘积
            x = x + g;  // 累加到 x 中
        }
```

```
        out4_buffer[w] = x;  // 将结果存入输出缓冲区
        w++;  // 更新输出指针
    }

    return TRUE;
}
```

详细解释

1. 函数参数

- `int *input1`：指向输入信号数据缓冲区 `inp1_buffer` 的指针（可能是信号 `x`）。
- `int *output2`：指向滤波器或脉冲响应数据缓冲区 `out2_buffer` 的指针（可能是滤波器响应 `h`）。
- `int *output4`：指向输出信号数据缓冲区 `out4_buffer` 的指针，用于存储卷积结果。

2. 正序卷积计算

```
for(;(m - y) > 0;)
{
    i = y;
    x = 0;
    z = 0;
    f = y;
    for(; i >= 0; i--) {
        g = input1[z] * output2[f];  // 计算输入信号和滤波器的乘积
        x = x + g;  // 累加到 x 中
        z++;  // 输入信号指针向后移动
        f--;  // 滤波器指针向前移动
    }
    *output4++ = x;  // 将结果存入输出缓冲区，并移动输出指针
    y++;  // 向后移动输出的指针
}
```

- 这部分代码实现了卷积的正序计算：从输入信号 `input1` 的起始位置开始，对其和滤波器 `output2` 进行加权乘积累加。
- `y` 是输出缓冲区的索引，它逐步增加，代表卷积结果的每个时间点。
- 内层 `for` 循环计算了 `input1` 和 `output2` 对应位置的乘积，并将其加到 `x` 中。然后，`z` 和 `f` 分别移动，`z` 表示输入信号的当前位置，`f` 表示滤波器的位置（从当前位置往前推）。

3. 倒序卷积计算

```
m = sk;
y = sk - 1;
w = m - zhy - 1;
for(; m > 0; m--) {
    y--;
    i = y;
    z = sk - 1;
    x = 0;
    f = sk - y;
    for(; i > 0; i--, z--, f++) {
        g = input1[z] * output2[f];  // 计算输入信号和滤波器的乘积
        x = x + g;  // 累加到 x 中
    }
```

```
    out4_buffer[w] = x;  // 将结果存入输出缓冲区
    w++;  // 更新输出指针
}
```

- 这部分是倒序的卷积计算。它与前面部分类似，但索引是倒着处理的，`z` 和 `f` 在内层 `for` 循环中是递减的。
- 该部分主要用于处理卷积后半部分，或者是滤波器反转后的卷积。

4. 卷积解释

- 在第一部分的正序计算中，`input1[z] * output2[f]` 是对输入信号 `input1` 和滤波器 `output2` 的点积操作（即加权求和）。
- 第二部分的倒序计算是为了完成卷积的另一半，确保从右到左的信号与滤波器的反向卷积得到正确的结果。
- 输出的卷积结果被存储在 `out4_buffer` 中，最终生成的 `output4` 即为卷积后的信号。

总结

`processing3` 是实现线性卷积操作的核心子程序。它通过两个部分：一个正序的卷积计算和一个倒序的卷积计算，来处理输入信号和滤波器之间的卷积。其过程包括：

1. 对输入信号和滤波器进行加权累加（点积操作），计算卷积的每个元素。
2. 使用正序和倒序两部分的计算来确保卷积结果的完整性。
3. 最终，结果存储在 `output4` 中，表示卷积后的输出信号。

这个过程是线性卷积的一种实现方式，常用于信号处理中的滤波操作。