```
typedef struct struct_model
{
    int k;         /* 基本関数の数k */
    int f_id[N];   /* 基本関数のid 1,...,NF*/
    double sol[N];/* モデルの求めたa1,...akの係数 */
    double err;    /* モデルの誤差 */
} struct_model;

enum result {TRUE, FALSE};


int f_in_model(struct_model model, int fi)
{
    int i;
    int found = FALSE;

    for (i =1; i<= model.k; i++)
    {
        if (fi == model.f_id[i])
        {
            found = TRUE;
            break;
        }
    }
    return found;
}

void copy_model(struct_model* p_current_m, struct_model* p_best_m)
{
    int j;

    (*p_best_m).err = (*p_current_m).err;
    (*p_best_m).k = (*p_current_m).k;

    for (j=1; j <= (*p_best_m).k; j++)
    {
        (*p_best_m).f_id[j] = (*p_current_m).f_id[j];
        (*p_best_m).sol[j]  = (*p_current_m).sol[j];
    }
}

void forward_step_wise_selection(int nf, double x[N], double y[N], int n,
 char* path, char* data)

{
    int i, k, j;
    struct_model current_m, best_m; /* 現行モデルと最高モデル */
    current_m.k =0;
    current_m.err = DBL_MAX; /* DBL_MAX <float.h> */
    best_m.err = DBL_MAX;

    char fname_hokan[200], fname_out[200];
    FILE *fp_out;
```

```
/* モデルの誤差と基本関数と求めた係数のファイル名を設定する  */
sprintf(fname_out,"%s/results/out_%s", path, data);
/* モデルの誤差と基本関数と求めた係数のファイルを開く  */
fp_out = fopen(fname_out, "w");


for (k=1; k <= nf; k++)     /* k basic functions */
{
    printf("--- k = %d ---\n", k);

    for (i=1; i<= nf; i++)
    {
        if (f_in_model(current_m, i) == FALSE)
        {
            /* 現在のモデルで関数fiを試す   */
            current_m.k = k;
            current_m.f_id[k] = i;

            /* モデルの係数を求める  */
            compute_model(n, k, x, y, current_m.f_id, current_m.sol);
            /* モデルの誤差を求める  */
            current_m.err = model_error(n, k , x, y, current_m.sol,
             current_m.f_id);

            /* 最高モデルを更新する */
            if (current_m.err < best_m.err)
                copy_model(&current_m, &best_m);

            /*  誤差と求めた基本関数の係数を出力 */
            printf("%d %f  ", i, current_m.err);
            for (j=1; j<=k; j++) {
                printf("%.3f*f%d ",current_m.sol[j], current_m.f_id[j]);
            }
            printf("\n");
        }
    }

    /*  モデルの誤差, モデルの基本関数と求めた係数を出力 */
    model_error_output(fp_out, k, best_m.err, best_m.k, best_m.sol,
     best_m.f_id);

    printf("k=%d 最高のモデル\n", best_m.k);
    printf(" 誤差:  %f\n", best_m.err);
    printf(" モデル: ");
    for (j=1; j <= best_m.k; j++) {
        printf("%.3f*f%d ",best_m.sol[j],best_m.f_id[j]);
    }
    printf("\n");
```

```
        /*  グラフを描くための準備（数表を出力）*/

        sprintf(fname_hokan,"%s/results/mk%d_est_%s", path, best_m.k, data);
      //printf("%s\n", fname_hokan);
        data_output(fname_hokan, n, best_m.k, x, y, best_m.sol, best_m.f_id);


        /*  現在モデルを更新する */
        copy_model(&best_m, &current_m);

    }
    /* ファイルを閉じる  */
    fclose(fp_out);

}
```