

# Summary of Lesson 18

## ex1.c

```
#include "ext.h"
#include "z_dsp.h"

// Finite Impulse Response(FIR) Filter (Lowpass)
//  $y[n] = (a_0 * x[n]) - (a_1 * x[n-1])$ 

typedef struct {
    t_pxobject b_ob;
    float last;
} t_ex1;

void ex1_float(t_ex1 *x, float input);
void ex1_dsp(t_ex1 *x, t_signal **sp, short *count);
t_int *ex1_perform(t_int *w);
void *ex1_new(t_symbol *s);

void *ex1_class;

void main()
{
    setup((t_messlist **)&ex1_class, (method)ex1_new, 0L, (short)sizeof(t_ex1),
    0L, 0L);
    address((method)ex1_dsp, "dsp", A_CANT, 0);
    dsp_initclass();
}

void ex1_dsp(t_ex1 *x, t_signal **sp, short *count)
{
    dsp_add(ex1_perform, 4, x, sp[0]->s_vec, sp[1]->s_vec, sp[0]->s_n);
}

t_int *ex1_perform(t_int *w)
{
    float *in, *out;
    long vs, i;
    t_ex1 *x;

    x = (t_ex1 *)w[1];
    in = (float *)w[2];
    out = (float *)w[3];
    vs = (long)w[4];

    for(i=0; i<vs; i++)
    {
```

```

    // y[n] = (a0 * x[n]) - (a1 * x[n -1])

    out[i] = 0.5 * in[i] + 0.5 * x->last;
    x->last = in[i];
}

return w+5;
}

void *ex1_new(t_symbol *s)
{
    t_ex1 *x;
    x = (t_ex1*)newobject(ex1_class);
    dsp_setup((t_pxobject*)x, 1);
    outlet_new((t_pxobject *)x, "signal");
    x->last = 0.0;
    return x;
}

```

---

## ex2.c

```

#include "ext.h"
#include "z_dsp.h"

// Finite Impulse Response(FIR) Filter with default values
// y[n] = (a0 * x[n]) + (a1 * x[n -1])

typedef struct {
    t_pxobject b_ob;
    float a0,a1;
    float last;
} t_ex2;

void ex2_float(t_ex2 *x, float input);
void ex2_dsp(t_ex2 *x, t_signal **sp, short *count);
t_int *ex2_perform(t_int *w);
void *ex2_new(t_symbol *s, float a0, float a1);

void *ex2_class;

void main()
{
    setup((t_messlist **)&ex2_class, (method)ex2_new, 0L, (short)sizeof(t_ex2),
    0L, A_FLOAT, A_FLOAT, 0L);
    addmess((method)ex2_dsp, "dsp", A_CANT,0);
    dsp_initclass();
}

void ex2_dsp(t_ex2 *x, t_signal **sp, short *count)
{
    dsp_add(ex2_perform, 4,x,sp[0]->s_vec,sp[1]->s_vec,sp[0]->s_n);
}

```

```
}
```

```
t_int *ex2_perform(t_int *w)
{
    float *in,*out;
    long vs,i;
    t_ex2 *x;

    x = (t_ex2 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long) w[4];

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = x->a0 * in[i] + x->a1 * x->last;
        x->last = in[i];
    }

    return w+5;
}
```

```
void *ex2_new(t_symbol *s, float a0, float a1)
{
    t_ex2 *x;
    x = (t_ex2 *) newobject(ex2_class);
    dsp_setup((t_pxobject *) x, 1);
    outlet_new((t_pxobject *) x, "signal");
    x->last = 0.0;
    x->a0 = a0;
    x->a1 = a1;
    return x;
}
```

---

### ex3.c

```
#include "ext.h"
#include "z_dsp.h"

// Finite Impulse Response(FIR) Filter Flexible
//  $y[n] = (a_0 * x[n]) + (a_1 * x[n - 1])$ 

typedef struct {
    t_pxobject b_ob;
    float a0,a1;
    float last;
} t_ex3;
```

```

void ex3_dsp(t_ex3 *x, t_signal **sp, short *count);
t_int *ex3_perform(t_int *w);
void *ex3_new(t_symbol *s, float a0, float a1);
t_int *ex3_perform_both(t_int *w);
t_int *ex3_perform_a0(t_int *w);
t_int *ex3_perform_a1(t_int *w);
t_int *ex3_perform_def(t_int *w);

void *ex3_class;

void main()
{
    setup((t_messlist **)&ex3_class, (method)ex3_new, 0L, (short)sizeof(t_ex3),
    0L, A_DEFFLOAT, A_DEFFLOAT, 0L);
    address((method)ex3_dsp, "dsp", A_CANT,0);
    dsp_initclass();
}

void ex3_dsp(t_ex3 *x, t_signal **sp, short *count)
{
    if(count[1] && count[2]) // both signal
        dsp_add(ex3_perform_both, 6,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[1]->s_vec, sp[2]->s_vec);
    else if(count[1] && (!count[2]))// only a0
        dsp_add(ex3_perform_a0, 5,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[1]->s_vec);
    else if((!count[1]) && count[2])// only a1
        dsp_add(ex3_perform_a1, 5,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[2]->s_vec);
    else // no connection
        dsp_add(ex3_perform_def, 4,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n);
}

t_int *ex3_perform_both(t_int *w)
{
    float *in,*out,*coef1,*coef2;
    long vs,i;
    t_ex3 *x;

    x = (t_ex3 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long)w[4];
    coef1 = (float *) (w[5]);
    coef2 = (float *) (w[6]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = coef1[i] * in[i] + coef2[i] * x->last;
    }
}

```

```

        x->last = in[i];
    }
    return w+7;
}

t_int *ex3_perform_a0(t_int *w)
{
    float *in,*out,*coef;
    long vs,i;
    t_ex3 *x;

    x = (t_ex3 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long) (w[4]);
    coef = (float *) (w[5]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = coef[i] * in[i] + x->a1 * x->last;
        x->last = in[i];
    }
    return w+6;
}

t_int *ex3_perform_a1(t_int *w)
{
    float *in,*out,*coef;
    long vs,i;
    t_ex3 *x;

    x = (t_ex3 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long) (w[4]);
    coef = (float *) (w[5]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = x->a0 * in[i] + coef[i] * x->last;
        x->last = in[i];
    }
    return w+6;
}

t_int *ex3_perform_def(t_int *w)
{
    float *in,*out;

```

```

    long vs,i;
    t_ex3 *x;

    x = (t_ex3 *)w[1];
    in = (float *)w[2];
    out = (float *)w[3];
    vs = (long)w[4];

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = x->a0 * in[i] + x->a1 * x->last;
        x->last = in[i];
    }

    return w+5;
}

void *ex3_new(t_symbol *s, float a0, float a1)
{
    t_ex3 *x;
    x = (t_ex3*)newobject(ex3_class);
    dsp_setup((t_pxobject*)x, 3);
    outlet_new((t_pxobject *)x, "signal");
    x->last = 0.0;
    x->a0 = a0;
    x->a1 = a1;
    return x;
}

```

---

#### ex4.c

```

#include "ext.h"
#include "z_dsp.h"

// Infinite Impulse Response(IIR) Filter
// ETA (exponential time average)
// y[n] = (a0 * x[n]) + (a1 * x[n -1])

typedef struct {
    t_pxobject b_ob;
    float a0,a1;
    float last;
} t_ex4;

void ex4_dsp(t_ex4 *x, t_signal **sp, short *count);
t_int *ex4_perform(t_int *w);
void *ex4_new(t_symbol *s, float a0, float a1);
t_int *ex4_perform_both(t_int *w);
t_int *ex4_perform_a0(t_int *w);
t_int *ex4_perform_a1(t_int *w);

```

```

t_int *ex4_perform_def(t_int *w);

void *ex4_class;

void main()
{
    setup((t_messlist **)&ex4_class, (method)ex4_new, 0L, (short)sizeof(t_ex4),
    0L, A_DEFFLOAT, A_DEFFLOAT, 0L);
    address((method)ex4_dsp, "dsp", A_CANT,0);
    dsp_initclass();
}

void ex4_dsp(t_ex4 *x, t_signal **sp, short *count)
{
    if(count[1] && count[2]) // both signal
        dsp_add(ex4_perform_both, 6,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[1]->s_vec, sp[2]->s_vec);
    else if(count[1] && (!count[2]))// only a0
        dsp_add(ex4_perform_a0, 5,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[1]->s_vec);
    else if((!count[1]) && count[2])// only a1
        dsp_add(ex4_perform_a1, 5,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n,
        sp[2]->s_vec);
    else // no connection
        dsp_add(ex4_perform_def, 4,x,sp[0]->s_vec,sp[3]->s_vec,sp[0]->s_n);
}

t_int *ex4_perform_both(t_int *w)
{
    float *in,*out,*coef1,*coef2;
    long vs,i;
    t_ex4 *x;

    x = (t_ex4 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long)w[4];
    coef1 = (float *) (w[5]);
    coef2 = (float *) (w[6]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = coef1[i] * in[i] + coef2[i] * x->last;
        x->last = out[i];
    }
    return w+7;
}

t_int *ex4_perform_a0(t_int *w)

```

```

{
    float *in,*out,*coef;
    long vs,i;
    t_ex4 *x;

    x = (t_ex4 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long) (w[4]);
    coef = (float *) (w[5]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = coef[i] * in[i] + x->a1 * x->last;
        x->last = out[i];
    }
    return w+6;
}

```

```

t_int *ex4_perform_a1(t_int *w)
{
    float *in,*out,*coef;
    long vs,i;
    t_ex4 *x;

    x = (t_ex4 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (long) (w[4]);
    coef = (float *) (w[5]);

    for(i=0;i<vs;i++)
    {
        // y[n] = (a0 * x[n]) + (a1 * x[n -1])
        out[i] = x->a0 * in[i] + coef[i] * x->last;
        x->last = out[i];
    }
    return w+6;
}

```

```

t_int *ex4_perform_def(t_int *w)
{
    float *in,*out;
    long vs,i;
    t_ex4 *x;

    x = (t_ex4 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);

```



```

vs = (long)w[4];

for(i=0;i<vs;i++)
{
    // y[n] = (a0 * x[n]) + (a1 * x[n -1])
    out[i] = x->a0 * in[i] + x->a1 * x->last;
    x->last = out[i];
}

return w+5;
}

void *ex4_new(t_symbol *s, float a0, float a1)
{
    t_ex4 *x;
    x = (t_ex4*)newobject(ex4_class);
    dsp_setup((t_pxobject*)x, 3);
    outlet_new((t_pxobject *)x, "signal");
    x->last = 0.0;
    x->a0 = a0;
    x->a1 = a1;
    return x;
}

```

---

#### ex5.c

```

#include "ext.h"
#include "z_dsp.h"
#include "buffer.h"

// buffer access index~

typedef struct {
    t_pxobject b_ob;
    t_buffer *buf;
    t_symbol *name;
} t_ex5;

void ex5_dsp(t_ex5 *x, t_signal **sp, short *count);
t_int *ex5_perform(t_int *w);
void *ex5_new(t_symbol *s);
t_int *ex5_perform(t_int *w);

void *ex5_class;

void main()
{
    setup((t_messlist **)&ex5_class, (method)ex5_new, 0L, (short)sizeof(t_ex5),
    0L, A_SYM, 0L);
    address((method)ex5_dsp, "dsp", A_CANT, 0);
    dsp_initclass();
}

```

```
}
```

```
void ex5_dsp(t_ex5 *x, t_signal **sp, short *count)
{
    t_buffer *b = NULL;
    b = (t_buffer *)(x->name->s_thing);

    if(b)
    {
        x->buf = b;
        dsp_add(ex5_perform, 4,x,sp[0]->s_vec,sp[1]->s_vec,sp[0]->s_n);
    }
    else
    {
        error("ex5: no buffer~ %s", x->name->s_name);
        x->buf = NULL;
    }
}
```

```
t_int *ex5_perform(t_int *w)
{
    float *in,*out;
    long vs,i;
    t_ex5 *x;
    long saveinuse;
    float *tab;
    int frames,index;
    t_buffer *b;

    x = (t_ex5 *)(w[1]);
    in = (float *)(w[2]);
    out = (float *)(w[3]);
    vs = (int)(w[4]);

    b = x->buf;
    saveinuse = b->b_inuse;
    b->b_inuse = 1;
    tab = b->b_samples;
    frames = b->b_frames;

    for(i=0;i<vs;i++)
    {
        index = (int)in[i];
        if(index < 0)
            index = 0;
        else if(index >= frames)
            index = frames-1;

        out[i] = tab[index];
    }
}
```

```

        b->b_inuse = saveinuse;

        return w+5;
    }

void *ex5_new(t_symbol *s)
{
    t_ex5 *x;
    x = (t_ex5*)newobject(ex5_class);
    dsp_setup((t_pxobject*)x, 1);
    x->name = s;
    outlet_new((t_pxobject *)x, "signal");
    return x;
}

```

---

#### ex6.c

```

#include "ext.h"
#include "z_dsp.h"
#include "buffer.h"

// buffer access index~ and peak finder

typedef struct {
    t_pxobject b_ob;
    t_buffer *buf;
    t_symbol *name;
    void *peakout;
} t_ex6;

void ex6_dsp(t_ex6 *x, t_signal **sp, short *count);
t_int *ex6_perform(t_int *w);
void ex6_bang(t_ex6 *x);
void *ex6_new(t_symbol *s);
t_int *ex6_perform(t_int *w);

void *ex6_class;

void main()
{
    setup((t_messlist **)&ex6_class, (method)ex6_new, 0L, (short)sizeof(t_ex6),
    0L, A_SYM, 0L);
    addmess((method)ex6_dsp, "dsp", A_CANT, 0);
    addbang((method)ex6_bang);
    dsp_initclass();
}

void ex6_bang(t_ex6 *x)
{
    t_buffer *b = x->buf;

```

```

if(!b)
{
    error("ex6 : no such table");
    return;
}

float max = 0.;
long peakpoint = -1;
long frames = b->b_frames;
int saveinuse = b->b_inuse;
float *tab = b->b_samples;
b->b_inuse = 1;

frames--;
while(frames--)
{
    if(max < tab[frames])
    {
        peakpoint = frames;
        max = tab[frames];
    }
}
b->b_inuse = saveinuse;
outlet_int(x->peakout, peakpoint);
}

void ex6_dsp(t_ex6 *x, t_signal **sp, short *count)
{
    t_buffer *b = NULL;
    b = (t_buffer *) (x->name->s_thing);

    if(b)
    {
        x->buf = b;
        dsp_add(ex6_perform, 4, x, sp[0]->s_vec, sp[1]->s_vec, sp[0]->s_n);
    }
    else
    {
        error("ex6: no buffer~ %s", x->name->s_name);
        x->buf = NULL;
    }
}

t_int *ex6_perform(t_int *w)
{
    float *in,*out;
    long vs,i;
    t_ex6 *x;

```

```

long saveinuse;
float *tab;
int frames,index;
t_buffer *b;

x = (t_ex6 *) (w[1]);
in = (float *) (w[2]);
out = (float *) (w[3]);
vs = (int) (w[4]);

    b = x->buf;
    saveinuse = b->b_inuse;
    b->b_inuse = 1;
    tab = b->b_samples;
    frames = b->b_frames;

    for(i=0;i<vs;i++)
    {
        index = (int) in[i];
        if(index < 0)
            index = 0;
        else if(index >= frames)
            index = frames-1;

        out[i] = tab[index];
    }
    b->b_inuse = saveinuse;

    return w+5;
}

void *ex6_new(t_symbol *s)
{
    t_ex6 *x;
    x = (t_ex6 *) newobject(ex6_class);
    dsp_setup((t_pxobject *) x, 1);
    x->name = s;
    x->peakout = intout(x);
    outlet_new((t_pxobject *) x, "signal");
    return x;
}

```

---

#### ex7.c

```

#include "ext.h"
#include "z_dsp.h"
#include "buffer.h"

// buffer access index~ and peak finder

typedef struct {
    t_pxobject b_ob;

```

```

    t_buffer *buf;
    t_symbol *name;
} t_ex7;

void ex7_dsp(t_ex7 *x, t_signal **sp, short *count);
t_int *ex7_perform(t_int *w);
void ex7_bang(t_ex7 *x);
void *ex7_new(t_symbol *s);
t_int *ex7_perform(t_int *w);

void *ex7_class;

void main()
{
    setup((t_messlist **)&ex7_class, (method)ex7_new, 0L, (short)sizeof(t_ex7),
0L, A_SYM, 0L);
    address((method)ex7_dsp, "dsp", A_CANT,0);
    addbang((method)ex7_bang);
    dsp_initclass();
}

void ex7_bang(t_ex7 *x)
{
    t_buffer *b = x->buf;

    if(!b)
    {
        error("ex7 : no such table");
        return;
    }

    float coef,max = 0.;
    long peakpoint = -1;
    long frames = b->b_frames -1;
    int saveinuse = b->b_inuse;
    float *tab = b->b_samples;
    b->b_inuse = 1;
    while(frames--)
    {
        if(max < tab[frames])
        {
            peakpoint = frames;
            max = tab[frames];
        }
    }
    coef = 1.0 / max;
    frames = b->b_frames -1;
    while(frames--)
    {
        max = tab[frames] *= coef;
    }
}

```

```

    }
    b->b_inuse = saveinuse;
}

void ex7_dsp(t_ex7 *x, t_signal **sp, short *count)
{
    t_buffer *b = NULL;
    b = (t_buffer *) (x->name->s_thing);

    if(b)
    {
        x->buf = b;
        dsp_add(ex7_perform, 4, x, sp[0]->s_vec, sp[1]->s_vec, sp[0]->s_n);
    }
    else
    {
        error("ex7: no buffer~ %s", x->name->s_name);
        x->buf = NULL;
    }
}

t_int *ex7_perform(t_int *w)
{
    float *in,*out;
    long vs,i;
    t_ex7 *x;
    long saveinuse;
    float *tab;
    int frames,index;
    t_buffer *b;

    x = (t_ex7 *) (w[1]);
    in = (float *) (w[2]);
    out = (float *) (w[3]);
    vs = (int) (w[4]);

    b = x->buf;
    saveinuse = b->b_inuse;
    b->b_inuse = 1;
    tab = b->b_samples;
    frames = b->b_frames;

    for(i=0;i<vs;i++)
    {
        index = (int) in[i];
        if(index < 0)
            index = 0;
        else if(index >= frames)
            index = frames-1;
    }
}

```

```
        out[i] = tab[index];
    }
    b->b_inuse = saveinuse;

    return w+5;
}

void *ex7_new(t_symbol *s)
{
    t_ex7 *x;
    x = (t_ex7*)newobject(ex7_class);
    dsp_setup((t_pxobject*)x, 1);
    x->name = s;
    outlet_new((t_pxobject *)x, "signal");
    return x;
}
```