

Summary of Lesson 12

Union

ex10_10.c

```
#include <stdio.h>

//an example of union

typedef union
{
    char initial;
    int age;
    double height;
}test_union;

int main (void) {

    test_union test1,test2,test3;
    printf("size of union %d\n",sizeof(test1));

    test1.initial = 'c';
    printf("%c\n",test1.initial);

    test2.age = 26;
    printf("%d\n",test2.age);

    test3.height = 167.24389520;
    printf("%f\n",test3.height);

    return 0;
}
```

Functions , introduced in this lesson

```
// create more int inlet
void intin (void *object, short index)

// establish a connection between nth int inlet and a function
void addinx (method mp, short inlet)
```

```

// create more float inlet
void floatin (void *object, short index)

// establish a connection between nth float inlet and a function
void addftx (method mp, short inlet)

//translation
Wrong :
void inlet_4 (void *owner, void *dst, t_symbol *in,
t_symbol *out);
Correct :
void inlet4 (void *owner, void *dst, char *in,
char *out);

// create bang outlet
Outlet *bangout (void *owner);

// send a bang from specified outlet
void *outlet_bang (Outlet *theOutlet);

// create int outlet
Outlet *intout (void *owner);

// send a int from specified outlet
void *outlet_int (Outlet *theOutlet, long n);

// create float outlet
Outlet *floatout (void *owner);

// send a float from specified outlet
void *outlet_float (Outlet *theOutlet, double f);

```

Structure and Union introduced in this lesson

```

union word /* union for packing any data type */
{
    long w_long;
    float w_float;
    t_symbol *w_sym;
    t_object *w_obj;
};

typedef struct atom /* an Atom that is a typed datum */
{
    short a_type; /* from the definitions above*/
    union word a_w;
} t_atom

typedef struct symbol
{
    char *s_name; /* name */

```

```
    struct object *s_thing; /* possible binding to an object */
} t_symbol, Symbol;
```

ex1.c

```
#include "ext.h"
// one more inlet

typedef struct {
    t_object b_ob;
} t_ex1;

void ex1_leftint(t_ex1 *x, long value);
void ex1_rightint(t_ex1 *x, long value);
void *ex1_new(void);

void *ex1_class;

void main()
{
    setup((t_messlist **)&ex1_class, (method)ex1_new, 0L, (short)sizeof(t_ex1),
    0L, 0L);
    addint((method)ex1_leftint );
    addinx((method)ex1_rightint, 1);
}

void ex1_leftint(t_ex1 *x, long value)
{
    post("I received %d from left inlet",value);
}

void ex1_rightint(t_ex1 *x, long value)
{
    post("I received %d from right inlet",value);
}

void *ex1_new(void)
{
    t_ex1 *x;
    x = (t_ex1*)newobject(ex1_class);
    intin(x, 1);
    return x;
}
```

ex2.c

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

```

//if you have three inlets

typedef struct {
    t_object b_ob;
} t_ex2;

void ex2_leftfloat(t_ex2 *x, float value);
void ex2_middelfloat(t_ex2 *x, float value);
void ex2_rightfloat(t_ex2 *x, float value);
void *ex2_new(void);

void *ex2_class;

void main()
{
    setup((t_messlist **)&ex2_class, (method)ex2_new, 0L, (short)sizeof(t_ex2),
    0L, 0L);
    addfloat((method)ex2_leftfloat );
    addftx((method)ex2_middelfloat, 1);
    addftx((method)ex2_rightfloat, 2);
}

void ex2_leftfloat(t_ex2 *x, float value)
{
    post("I received %f from left inlet",value);
}

void ex2_middelfloat(t_ex2 *x, float value)
{
    post("I received %f from middle inlet",value);
}

void ex2_rightfloat(t_ex2 *x, float value)
{
    post("I received %f from right inlet",value);
}

void *ex2_new(void)
{
    t_ex2 *x;
    x = (t_ex2*)newobject(ex2_class);
    // order is important
    floatin(x, 2);
    floatin(x, 1);
    return x;
}

```

ex3.c

// keyword "anything" and check typeof input manually

```

#include "ext.h"

typedef struct {
    t_object b_ob;
    long value;
} t_ex3;

void ex3_anything(t_ex3 *x, t_symbol *message, short argc, t_atom *argv);
void *ex3_new(void);

void *ex3_class;

void main()
{
    setup((t_messlist **)&ex3_class, (method)ex3_new, 0L, (short)sizeof(t_ex3),
    0L, 0L);
    addmess((method)ex3_anything, "anything", A_GIMME, 0 );
}

void ex3_anything(t_ex3 *x, t_symbol *message, short argc, t_atom *argv)
{
    int i;
    t_atom *at;

    post("The message is %s",message->s_name);
    post("%d more elements are attached to it",argc);

    for(i = 0;i< argc;i++)
    {
        at = argv+i;

        switch(at->a_type)
        {
            case A_LONG:
                post("No.%d type:int content:%d",i,argv[i].a_w.w_long);
                break;
            case A_FLOAT:
                post("No.%d type:float content:%f",i,argv[i].a_w.w_float);
                break;
            case A_SYM:
                post("No.%d type:symbol
content:%s",i,argv[i].a_w.w_sym->s_name);
                break;
        }
    }
}

void *ex3_new(void)
{
    t_ex3 *x;

```

```

    x = (t_ex3*)newobject(ex3_class);
    return x;
}

```

ex4.c

```

// anything and list
#include "ext.h"

typedef struct {
    t_object b_ob;
    long value;
} t_ex4;

void ex4_list(t_ex4 *x, t_symbol *message, short argc, t_atom *argv);
void *ex4_new(void);

void *ex4_class;

void main()
{
    setup((t_messlist **)&ex4_class, (method)ex4_new, 0L, (short)sizeof(t_ex4),
    0L, 0L);
    addmess((method)ex4_list, "list", A_GIMME, 0 );
}

void ex4_list(t_ex4 *x, t_symbol *message, short argc, t_atom *argv)
{
    int i;
    t_atom *at;

    // post("The message is %s",message->s_name); un imported should be
    ignored!!
    post("The list consist of %d elements",argc);

    for(i = 0;i< argc;i++)
    {
        at = argv+i;

        switch(at->a_type)
        {
            case A_LONG:
                post("No.%d type:int content:%d",i,argv[i].a_w.w_long);
                break;
            case A_FLOAT:
                post("No.%d type:float content:%f",i,argv[i].a_w.w_float);
                break;
            case A_SYM:
                post("No.%d type:symbol
content:%s",i,argv[i].a_w.w_sym->s_name);

```

```

        break;
    }
}

void *ex4_new(void)
{
    t_ex4 *x;
    x = (t_ex4*)newobject(ex4_class);
    return x;
}

```

ex5.c

```

#include "ext.h"

typedef struct {
    t_object b_ob;
    long value;
} t_ex5;

void ex5_list(t_ex5 *x, t_symbol *message, short argc, t_atom *argv);
void *ex5_new(void);

void ex5_loadbang (t_ex5 *x);
void ex5_assist (t_ex5 *x, void *box, long msg, long arg, char *dstString);

void *ex5_class;

void main()
{
    setup((t_messlist **)&ex5_class, (method)ex5_new, 0L, (short)sizeof(t_ex5),
    0L, 0L);
    address ((method)ex5_loadbang, "loadbang", A_CANT, 0);
    address ((method)ex5_assist, "assist", A_CANT, 0);
}

void ex5_loadbang (t_ex5 *x)
{
    post("I was loaded.");
}

void ex5_assist (t_ex5 *x, void *box, long msg, long arg, char *dstString)
{
    switch(msg)
    {
        case ASSIST_INLET:
            if(arg == 0)

```

```

        {
            sprintf(dstString,"you are pointing my leftmost inlet");
        }
        break;
    }
}

void *ex5_new(void)
{
    t_ex5 *x;
    x = (t_ex5*)newobject(ex5_class);
    return x;
}

```

ex6.c

```

#include "ext.h"

typedef struct {
    t_object b_ob;
} t_ex6;

void ex6_hallo(t_ex6 *x);
void *ex6_new(void);

void *ex6_class;

void main()
{
    setup((t_messlist **)&ex6_class, (method)ex6_new, 0L, (short)sizeof(t_ex6),
    0L, 0L);
    addmess((method)ex6_hallo, "hallo", 0 );
}

void *ex6_new(void)
{
    t_ex6 *x;
    x = (t_ex6*)newobject(ex6_class);
    inlet4(x,x,"hello","hallo");
    //inlet4(x,x,"bye","hallo");

    return x;
}

void ex6_hallo(t_ex6 *x)
{
    post("Hallo");
}

```

ex7.c


```

// outlet
#include "ext.h"

typedef struct {
    t_object b_ob;
    void *out1;
} t_ex7;

void ex7_bang(t_ex7 *x);
void *ex7_new(void);

void *ex7_class;

void main()
{
    setup((t_messlist **)&ex7_class, (method)ex7_new, 0L, (short)sizeof(t_ex7),
    0L, 0L);
    addbang((method)ex7_bang);
}

void *ex7_new(void)
{
    t_ex7 *x;
    x = (t_ex7*)newobject(ex7_class);
    x->out1 = bangout(x);
    return x;
}

void ex7_bang(t_ex7 *x)
{
    outlet_bang(x->out1);
}

```

ex8.c

```

#include "ext.h"

// assignment easy uzi

typedef struct {
    t_object b_ob;
    void *out1;
} t_ex8;

void ex8_int(t_ex8 *x,int num);
void *ex8_new(void);

void *ex8_class;

```

```

void main()
{
    setup((t_messlist **)&ex8_class, (method)ex8_new, 0L, (short)sizeof(t_ex8),
0L, 0L);
    addint((method)ex8_int);
}

void *ex8_new(void)
{
    t_ex8 *x;
    x = (t_ex8*)newobject(ex8_class);
    x->out1 = bangout(x);
    return x;
}

void ex8_int(t_ex8 *x,int num)
{
    int i;
    for(i = 0; i < num;i++)
    {
        outlet_bang(x->out1);
    }
}

```

ex9.c

```

#include "ext.h"

// easy counter

typedef struct {
    t_object b_ob;
    long count;
    void *out1;
} t_ex9;

void ex9_bang(t_ex9 *x);
void ex9_reset(t_ex9 *x);

void *ex9_new(void);

void *ex9_class;

void main()
{
    setup((t_messlist **)&ex9_class, (method)ex9_new, 0L, (short)sizeof(t_ex9),
0L, 0L);
    addmess((method)ex9_reset,"reset",0L);
    addbang((method)ex9_bang);
}

```

```

void *ex9_new(void)
{
    t_ex9 *x;
    x = (t_ex9*)newobject(ex9_class);
    x->out1 = intout(x);
    x->count = 0;

    return x;
}

void ex9_bang(t_ex9 *x)
{
    x->count++;
    outlet_int(x->out1, x->count);
}

void ex9_reset(t_ex9 *x)
{
    x->count = 0;
    outlet_int(x->out1, x->count);
}

```

ex10.c

```

#include "ext.h"

// easy plus

typedef struct {
    t_object b_ob;
    long current;
    void *out1;
} t_ex10;

void ex10_calc(t_ex10 *x, long input);
void ex10_store(t_ex10 *x, long input);

void *ex10_new(void);

void *ex10_class;

void main()
{
    setup((t_messlist **)&ex10_class, (method)ex10_new, 0L,
    (short)sizeof(t_ex10), 0L, 0L);
    addint((method)ex10_calc);
    addinx((method)ex10_store, 1);
}

```

```
void *ex10_new(void)
{
    t_ex10 *x;
    x = (t_ex10*)newobject(ex10_class);
    intin(x,1);
    x->out1 = intout(x);
    x->current = 0;

    return x;
}

void ex10_calc(t_ex10 *x,long input)
{
    outlet_int( x->out1, x->current + input );
}

void ex10_store(t_ex10 *x,long input)
{
    x->current = input;
}
```