

# Summary of Lesson 8

## Pointer<1>

- . All variable, array, structure variable have **own address** on the memory
- . you can get the address using **&**
- . Pointer points an **address of an another variable**
- . To declare the pointer use \* before name
- . Array name returns the address of **first index**
- . You can access a variable and change it from **a different function**
- . To refer members of a sturcture variable use -> instead of .

---

### lesson8\_1.c

```
#include <stdio.h>

int main (void) {

    int x = 3;

    printf("the Content of x is %d\n", x);
    printf("the Address of x is %u\n", &x);
    printf("the Size of x is %d bytes\n", sizeof(x));

    return 0;
}
```

---

### lesson8\_2.c

```
#include <stdio.h>

int main (void) {

    int x = 3;
    int *xptr = &x;

    printf("the Content of x is %d\n", *xptr);
    printf("the Address of x is %u\n", xptr);

    return 0;
}
```

---

### lesson8\_3.c

```
#include <stdio.h>

int main (void) {
```

```

int x = 3;
int *xptr = &x;

printf("The content of x is %d\n",x);
printf("The address of x is %u\n",&x);
printf("The content of xptr is %u\n",xptr);
printf("The address of xptr is %u\n",&xptr);
printf("xptr tells us the content of x is %d\n",*xptr);
printf("xptr tells us the address of x is %u\n",xptr);

return 0;
}

```

---

#### lesson8\_4.c

```

#include <stdio.h>

int main (void) {

    int i;
    int xarray[5] = {1,2,4,8,16};
    int *xptr = xarray;

    printf("The Address of index 0 of the array is %u\n",xarray);
    printf("and its content is %d\n",*xarray);

    printf("The Address of index 1 of the array is %u\n",xarray+1);
    printf("and its content is %d\n",*(xarray+1));

    printf("The Address of index 0 of the array is %u\n",xptr);
    printf("and its content is %d\n",*xptr);

    printf("The Address of index 2 of the array is %u\n",xptr+2);
    printf("and its content is %d\n",*(xptr+2));
    printf("and its content is %d\n",xptr[2]);

    printf("print all of them,using pointer\n");
    for(i = 0;i<5;i++)
    {
        printf("index %d is %d\n",i,*(xptr+i));
    }

    return 0;
}

```

---

#### lesson8\_5.c

```

#include <stdio.h>

```

```

int main (void) {

    int i;
    int xarray[5] = {1,2,4,8,16};
    int *xptr = xarray;

    for(i = 0;i<5;i++)
    {
        printf("index %d is %d\n",i,xptr[i]); // square bracket
    }

    for(i = 0;i<5;i++)
    {
        printf("index %d is %d\n",i,*(xptr++));
        // increment pointer directly
    }

    return 0;
}

```

---

#### lesson8\_6.c

```

#include <stdio.h>

void test(int *ptr);

int main (void) {

    int x = 3;
    test(&x);
    printf("%d",x);
    return 0;
}

void test(int *ptr)
{
    *ptr = 5;
}

```

---

#### lesson8\_7.c

```

#include <stdio.h>

void capitalizer(char *string,int len);

int main (void) {

    char name[] = "chikashi";
    capitalizer(name,8);
}

```

```

    printf("%s",name);
    return 0;
}

void capitalizer(char *string,int len)
{
    int i;
    int gap = (int)'A' - 'a';

    for(i = 0;i<len;i++)
    {
        *(string+i) = *(string+i) += gap;
    }
}

```

---

### lesson8\_8.c

```

#include <stdio.h>

typedef struct
{
    float height;
    float width;
}Triangle;

float triangleArea(Triangle *tri);

int main (void) {

    float area;
    Triangle aTriangle = {3.0,5.5};
    area = triangleArea(&aTriangle);
    printf("%.2f",area);

    return 0;
}

float triangleArea(Triangle *tri)
{
    return tri->height * tri->width; // use arrow operator for reference
}

```