

# Summary of Lesson 9

## Pointer<2>

- with malloc you can allocate memory **dynamically**
- you can refer the values in the dynamically allocated memory like **an array**
- calloc() is similar to malloc() but fill all contents with **zero**
- realloc() try to **resize** already allocated memory
- you can refer all pointer of variables with void\* but you can not refer it directly
- before reference you have to **cast** the pointer to appropriate type
- you can point also a **function** and **invoke** it with direct operator

---

### lesson9\_1.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int *ptr;
    ptr = (int*)malloc(sizeof(int));
    *ptr = 3;

    printf("the value stored at dynamically allocated memory is %d\n",*ptr);

    free(ptr);
    // you must always free after you finish using it otherwise it leaks.

    return 0;
}
```

---

### lesson9\_2.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i;
    int *ptr;
    ptr = (int*)malloc(sizeof(int) * 5);

    for(i = 0;i<5;i++)
    {
        ptr[i] = i*i;
    }

    for(i = 0;i<5;i++)
```

```
    {
        printf("ptr[%d] is %d\n",i,ptr[i] );
    }

    free(ptr);
    return 0;
}
```

---

### lesson9\_3.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i;
    int *ptr;
    ptr = (int*)calloc(10, sizeof(int));
    for(i = 0;i < 10;i++)
    {
        printf("%d",ptr[i]);
    }
    free(ptr);
    return 0;
}
```

---

### lesson9\_4.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i;
    int *ptr;
    ptr = (int*)malloc(sizeof(int));
    *ptr = 3;
    // you can store only one value

    ptr = realloc(ptr,sizeof(int) * 5);
    // now up to 5

    ptr[4] = 4;
    for(i = 0;i < 5;i++)
    {
        printf("%d ",ptr[i]);
    }
    free(ptr);
    return 0;
}
```

---

### lesson9\_5.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    long i = 3;
    void *ptr;

    ptr = (void*)&i; // it can point anything

    //printf("%d",*ptr); you can not refer directly

    printf("%d",*((long*)ptr)); // cast pointer and refer

    return 0;
}
```

---

### lesson9\_6.c

```
#include <stdio.h>

int test(void);

int main(void)
{
    printf("the address of test function is %u",test);

    return 0;
}

int test(void)
{
    puts("test function");
    return 0;
}
```

---

### lesson9\_7.c

```
#include <stdio.h>

int test(void);

int main(void)
{
    int (*fptr)() = test;
    (*fptr)(); // call test function from pointer
}
```

```
        return 0;
    }

int test(void)
{
    puts("test function");
    return 0;
}
```

---

### lesson9\_8.c

```
#include <stdio.h>
```

```
char* capitalizer(char* input, int len);
```

```
int main(void)
{
    char name[] = "chikashi";
    char *(*fptr)() = capitalizer;
    // declare pointer for function, which returns a pointer
    printf("%s", (*fptr)(name,8));
    return 0;
}
```

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

    for (i = 0; i < len; i++)
    {
        input[i] += gap;
    }

    return input;
}
```