

Programming Study Pointer

Sungwoo Nam
2019.3.11

Pointer Usage

```
int a = 1234;  
int *pa = &a;    // pa is pointer of a  
*pa = 1111;      // assignment on *pa will change a also  
  
int &ra = a;      // ra is reference of a  
ra = 4321;        // assignment on ra will change a also
```

Function Return using reference and pointer

```
int AddAndMultiply(int a, int b, int& mult)
{
    mult = a * b;
    return a + b;
}
```

```
int AddAndMultiplyV3(int a, int b, int* mult)
{
    *mult = a * b;
    return a + b;
}
```

```
int mult = 0;
int sum = AddAndMultiply(1, 2, mult);
sum = AddAndMultiplyV3(1, 2, &mult);
```

Function Return using struct








```
struct AddMult  
{  
    int Add;  
    int Mult;  
};
```

```
AddMult AddAndMultiplyV3(int a, int b)  
{  
    AddMult r;  
    r.Add = a + b;  
    r.Mult = a * b;  
    return r;  
}
```

```
AddMult r = AddAndMultiplyV3(1, 2);  
sum = r.Add;  
mult = r.Mult;
```

Const Usage

```
const int* cpa = &a;  
*cpa = 42;           // compile error : cpa is const pointer  
cpa = &b;             // but you can make it point others  
  
const int& cra = a;  
cra = 42;             // compile error : cra is const reference  
  
int* const pca = &a;  
*pca = 42;            // no problem  
*pca = &b;            // compile error : pointer itself is const
```

 a	42
▷  pa	0x00b9f4a0 {42}
 ra	42
 b	2222
▷  cpa	0x00b9f48c {2222}
 cra	42
▷  pca	0x00b9f4a0 {42}

Const on class method

```
struct AddData
{
    int A;
    int B;
    mutable int C = 0;  // ok to be changed on const method

    int AddAll() const  // This doesn't change member variable
    {
        if (C == 0) { C = A + B; }
        return C;
    }
};
```

Endianness

```
uint32_t *pa = new uint32_t[4];  
pa[0] = 0x12345678; // note the little-endian  
pa[1] = 0x87654321;  
pa[2] = 0x10101010;  
pa[3] = 0xCAFEFEEF;  
  
*pa = 0xFFFFFFFF;  
*(pa + 1) = 0xEEEEEEEE;  
  
delete[] pa;
```

Watch 1	
Name	Value
pa	0x00d4c2f0 {0x12345678}
pa,4	0x00d4c2f0 {0x12345678, 0xcdcdcdcd, 0xcdcdcdcd, 0xcdcdcdcd}

Memory	
Address:	0x00D4C2F0
0x00D4C2F0	78 56 34 12 cd cd cd cd cd cd cd cd cd cd cd fd fd fd f
0x00D4C30D	fe ee fe 00 00 00 00 00 00 00 00 c5 6f 7b 97 39 5d 00 00 0
0x00D4C32A	ee fe ee fe ee fe ee fe ee fe ee fe ee fe ee fe ee fe ee f

x64 application

```
uint32_t *pa = new uint32_t[4]; // note pa is 64 bit length
pa[0] = 0x12345678;
pa[1] = 0x87654321;
pa[2] = 0x10101010;
pa[3] = 0xCAFEFEEF;

*pa = 0xFFFFFFFF;
*(pa + 1) = 0xEEEEEEEE;

delete[] pa;
```

Name	Value
▷ pa	0x0000022d51778890 {0x12345678}
▷ pa,4	0x0000022d51778890 {0x12345678, 0xcdcdcdcd, 0xcdcdcdcd, 0xcdcdcdcd}

Memory 1															
Address: 0x0000022D51778890															
0x0000022D51778890	78	56	34	12	cd	cd	cd	cd	cd	cd	cd	cd	cd	cd	fd
0x0000022D517788AB	ab	ab	ab	ab	ab	ab	ab	ab	ab	ee	fe	ee	fe	ee	fe
0x0000022D517788C6	00	00	00	00	00	00	00	00	00	ee	fe	ee	fe	ee	fe
0x0000022D517788E1	8f	77	51	2d	02	00	00	70	23	70	51	2d	02	00	00

Struct pack

```
struct Foo
{
    uint32_t a;        // 4 Bytes
    char b;            // 2 Bytes
    uint16_t c;        // 2 Bytes
    bool d;            // 4 Bytes
    const char * e;    // 4 Bytes
};
```

```
Foo foo;
```

```
Foo* pfoo = &foo;
```

```
foo = { 0x12345678, 'a', 0xCAFE, true, nullptr };
```

Name	Value
▷ foo	{a=0xffffffff b=0xcc '?' c=0xcccc ...}
▷ pfoo	0x0166fb24 {a=0xffffffff b=0xcc '?' c=0xcccc ...}

Memory 1	
Address:	0x0166FB24
0x0166FB24	cc 70 c2 6c
0x0166FB24	78 56 34 12 61 cc fe ca 01 cc cc cc 00 00 00 00 cc cc cc cc 70 c

Function Pointer

```
int TestBar(char a) { return a + 42; }  
int TestToo(char a) { return a + 22; }  
  
typedef int(*FooFn)(char);  
FooFn f = TestBar;  
int ret = f('a');  
f = TestToo;  
ret = f('b');
```

Function Pointer using STL

```
int TestBar(char a) { return a + 42; }  
int TestToo(char a) { return a + 22; }  
  
using FooFn = std::function<int(char)>;  
FooFn fn = TestBar;  
int ret = fn('a');  
fn = TestToo;  
ret = fn('b');
```