# 【C++】Day five(2)

|  |  |
|---|---|
| Class | C++ |
| Date | @November 22, 2021 |
| Material |  |
| Series Number |  |
| Summary |  |

## 【Ch2】Pointers

### 2.3.2 Pointers

A pointer is a compound type that "points to" another type.

Reference vs. Pointers

- A pointer is an object in its own right.
- Pointers can be assigned and copied
- A single pointer can point to several different object over its lifetime.
- A pointer need to be initialized at the time it is defined.
- Pointers defined at block scope have undefined value if they are not initialized.

We define a pointer type by writing a declarator of the form `*d`, where `d` is the name being defined. The `*` must be repeated for each pointer variable.

```
int *ip1, *ip2;
```

A pointer holds the address of another object. We get the address of an object by using the address-of operator `&`.

```
int val = 3;
int *pVal = &val;
```

*Note: Because references are not object, they don't have addresses. Hence, we may not define a pointer to a reference.*

The types of the pointer and the object to which it points must match:

```
double dValal;
double *pd = &dVal; //ok: initializer is the address of a double
int *pi = pd; //error: types of pi and pd differ
pi = &dVal; //error: assigning the address of a double to a pointer to int
```

The types must match because the type of the pointer is used to infer the type of the object to which the pointer points. If a pointer addressed an object of another type, operations performed on the underlying object would fail.

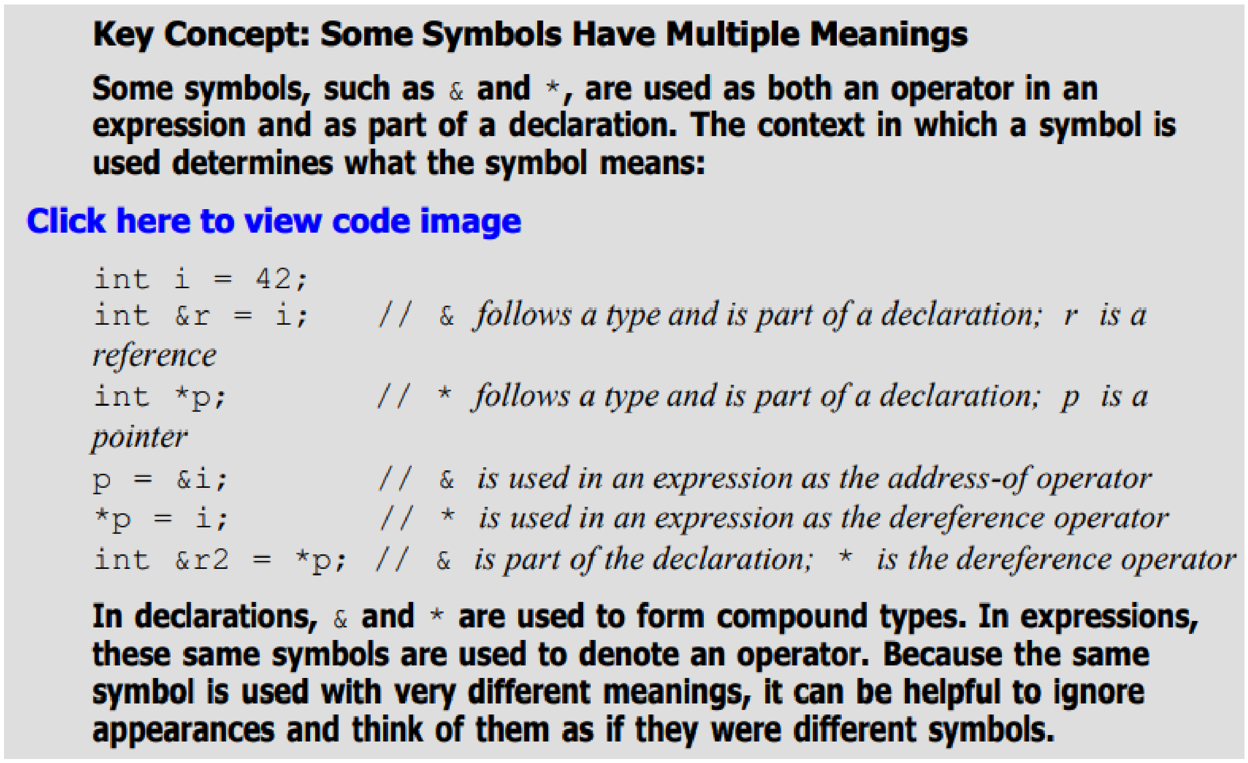
*Pointer Value*

The value stored in a pointer can be in one of four states:

1. It can point to an object.
2. It can point to the location just immediately past the end of an object.
3. It can be a null pointer, indicating that it is not bound to any object.
4. It can be invalid; values other than the preceding three are invalid.

*Note: We may dereference only a valid pointer that points to an object.*

The meaning of `&` and `*` could be different when used in expressions and declarations.



**Key Concept: Some Symbols Have Multiple Meanings**

**Some symbols, such as `&` and `*`, are used as both an operator in an expression and as part of a declaration. The context in which a symbol is used determines what the symbol means:**

Click here to view code image

```
int i = 42;
int &r = i;     // & follows a type and is part of a declaration; r is a reference
int *p;         // * follows a type and is part of a declaration; p is a pointer
p = &i;         // & is used in an expression as the address-of operator
*p = i;         // * is used in an expression as the dereference operator
int &r2 = *p;   // & is part of the declaration; * is the dereference operator
```

**In declarations, `&` and `*` are used to form compound types. In expressions, these same symbols are used to denote an operator. Because the same symbol is used with very different meanings, it can be helpful to ignore appearances and think of them as if they were different symbols.**

A null pointer does not point to any object. Code can check whether a pointer is null before attempting to use it. There are several ways to obtain a null pointer:

```
int *p1 = nullptr; //equivalent to int *p1 = 0;
int *p2 = 0; //directly initializes p2 from the literal constant 0
//must #include cstdlib
int *p3 = NULL;
```

The most direct approach is to initialize the pointer using the literal `nullptr`, which was introduced by the new standard. `nullptr` is a literal that has a special type that can be converted to any other pointer type.

Alternatively, we can initialize a pointer to the literal 0, as we do in the definition of p2.

Older programs sometimes use a preprocessor variable names NULL, which the `cstdlib` header defines as 0.

It is illegal to assign an `int` variable to a pointer, even if the variable's value happens to be 0;

```
int num = 0;
int *ptr = num; //error: cannot assign an int to a pointer.
```

### *Advice: initialize all Pointers*

- If possible ,define a pointer only after the object to which it should point has been defined.
- If there is no object to bind to a pointer, then initialize the pointer to `nullptr` or zero. That way, the program can detect the pointer does not point to an object.

### *void* pointers*

The type `void*` is a special pointer type that can hold the address of any object. Like any other pointer, a void* pointer holds an address, but the type of the object at that address is unknown.

```
double obj = 3.14, *pd = &obj;
void *pv = &obj; //obj can be an object of any type
pv = pd;
```

There are only a limited number of things we can do with a void* pointer:

1. We can compare it to another pointer.
2. We can pass it to or return it from a function.
3. we can assign it to another void* pointer.

We cannot use a void* to operate on the object it addresses as we don't know that object's type, and the type determines what operations we can perform on the object.