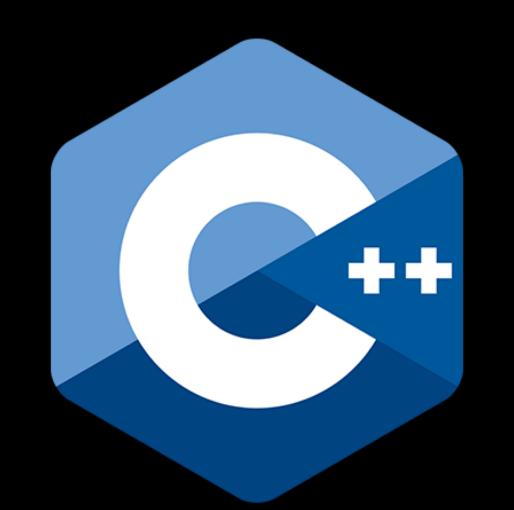
سلسلة تعلم البرمجة بلغة ++2 الحديثة

Learn Modern C++ Programming Course

إعداد المهندس أحمد الديب



#6: Value Categories lvalue & rvalue

Ivalue & rvalue

- an Ivalue is an expression that refers to an object (contiguous region of storage).
- rvalue means a value that is not an Ivalue.
- Ivalue originally meant "something that can be on the left-hand side of an assignment." Not every Ivalue may be used on the left-hand side of an assignment; an Ivalue can refer to a constant
- An Ivalue that has not been declared const is often called a modifiable lvalue.
- Address of an Ivalue may be taken by built-in address-of operator &...

Example

```
int var1 = 4;
int *pvar = &var1;
int var2 = var1 + 4;

&var1 = 10; // error: lvalue required as left operand of assignment

(var1 = 4) = 10;
(var1 + 4) = 10; // error: lvalue required as left operand of assignment)
```

Example 2

```
int var = 20;
int& fun1() { return var; }
int fun2() { return var; }

int main() {
   int x1 = fun1(); // fun1() is lvalue
   fun1() = 10;

   int x2 = fun2(); // fun2() is rvalue
   fun2() = 10; // error: lvalue required as left operand of assignment
   return 0;
}
```

C++11 General rule

- Two properties that matter for an object when it comes to addressing, copying, and moving.
- Has identity: The program has the name of, pointer to, or reference to the object.
- Movable: The object may be moved from; move semantics e.g. pointer to dynamically allocated memory.

More categories

| Has identity | Movable | category | category |
|--------------|---------|----------|--------------------|
| Y | N | Ivalue | generalized Ivalue |
| V | Y | xvalue | |
| | | | rvalue |
| N | Y | prvalue | |

```
42 // movable (x = 42) and not identifiable \rightarrow rvalue, prvalue a + b // movable (x = a + b) and not identifiable \rightarrow rvalue, prvalue ++a // ++a = 20 is OK \rightarrow identifiable and not movable \rightarrow glvalue, lvalue a++ // a++ = 20 is ERROR \rightarrow movable and not identifiable \rightarrow rvalue, prvalue
```

Pre-increment and pre-decrement operators increments or decrements the value of the object and returns a reference to the result.

Post-increment and post-decrement creates a copy of the object, increments or decrements the value of the object and returns the copy from before the increment or decrement.

Thankyou