@[toc]

中期第1次直播 typedef

typedef

定义方法：

```
typedf  struct student {
   int num ;
   char name[20] ;
   char sex ;
}  stu , * pstu ;
```

- `stu` 代表 `struct student` （给结构体类型起别名）
- `* pstu` 代表 `struct studen *` （给结构体指针变量起别名）
- `typedef int INTEGER` 起别名的作用在于**代码即注释**

C++ '&' 符的运用

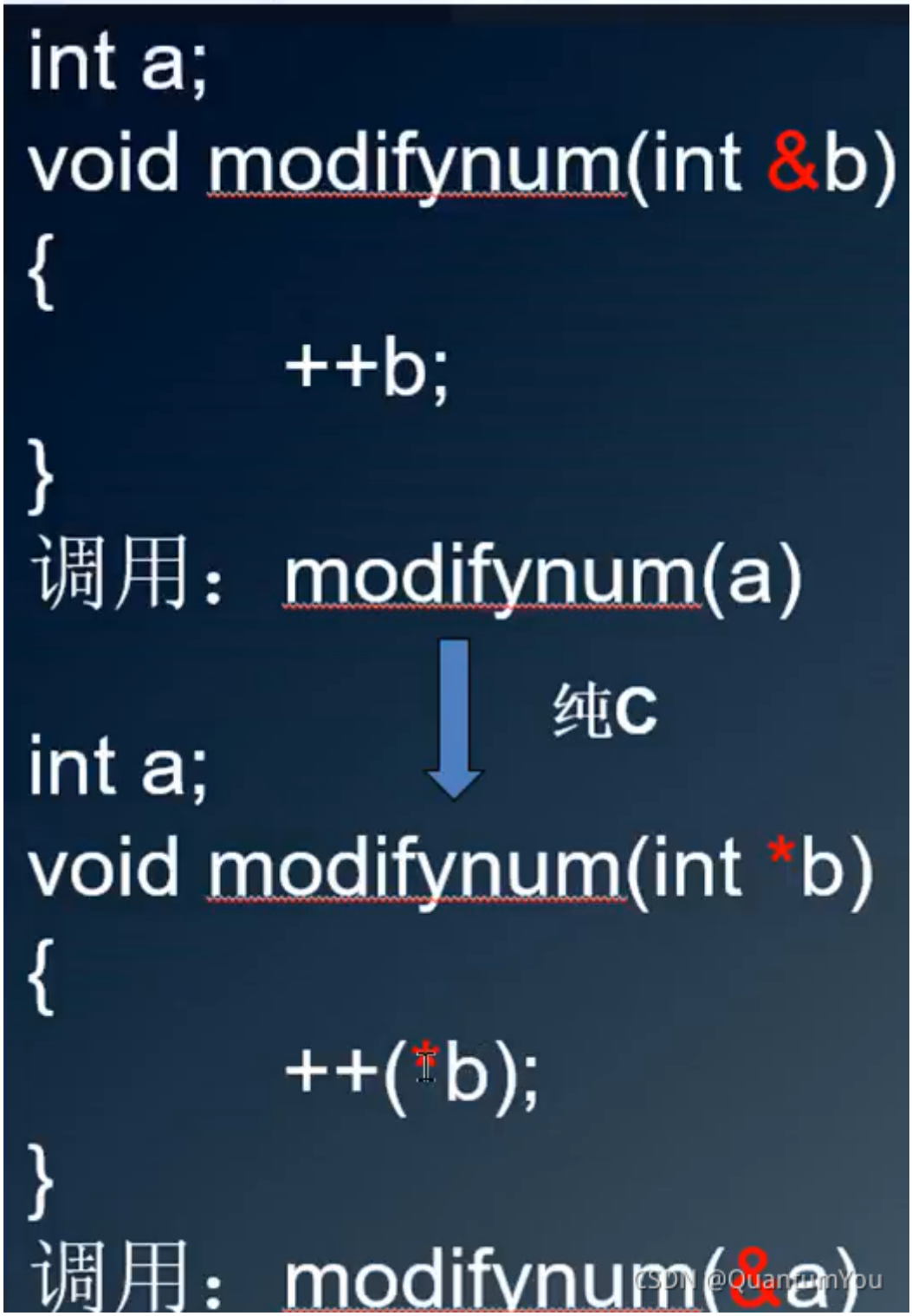
- 把`&`写到形参的位置是C++的语法，称为引用

```
#include  <stdio.h>

void modify_num(int &b){
   b = b+1 ;
}

int main(){
  int a = 10 ;
  modify_num(a) ;
  printf("a=%d\n",a) ;
  return 0 ;
}
```

- 如果将 引用改为纯C 的写法如下：



运用引用操作指针

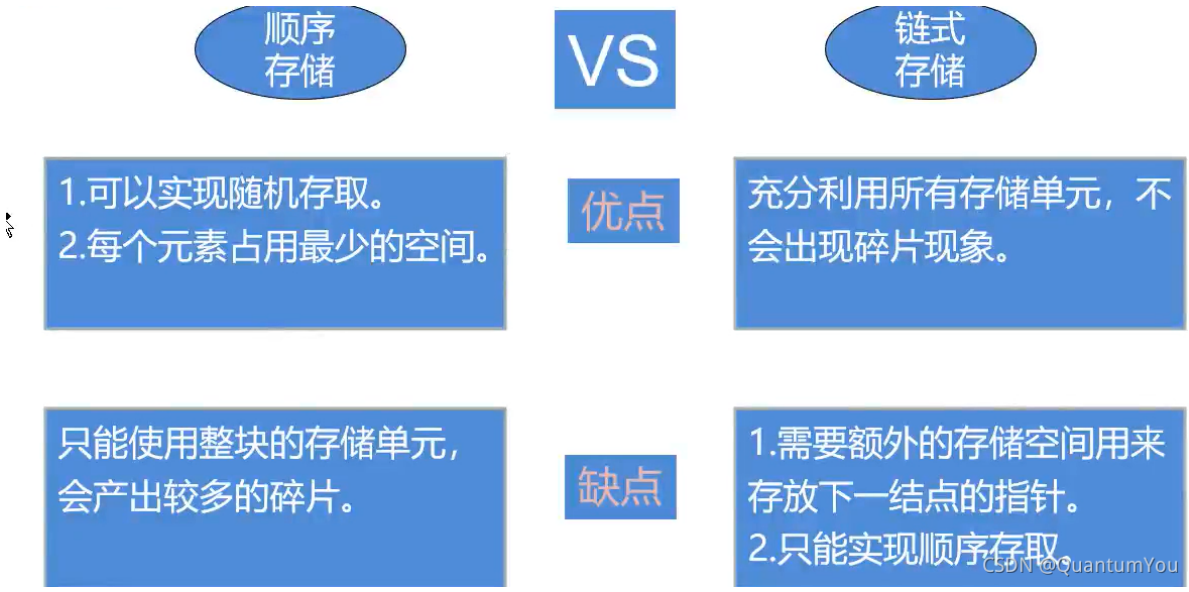
```
void modify_pointer(int*& p)//在子函数内操作p和主函数操作p手法一致
{
    p = (int*)malloc(20);
    p[0] = 5;
}
int main()
{
    int a = 10;
    modify_num(a);
    printf("a=%d\n", a);
    int* p = NULL;
    modify_pointer(p);
    printf("p[0]=%d\n", p[0]);
    return 0;
}
```


中期第2次直播 逻辑/物理 结构

逻辑结构：集合结构、线性结构、树形结构、图形结构

物理结构：顺序存储、链式存储、索引存储、散列存储

顺序存储与随机存储对比



时间复杂度、空间复杂度

- 时间复杂度指算法中所有语句的频度（执行次数）之和。
- 空间复杂度指算法运行过程中所使用的辅助空间的大小。

线性表的顺序存储及其原理实现

优点

- 可以随机存取（根据表头元素地址和元素序号）表中任意一个元素。
- 存储密度高，每个结点只存储数据元素。

缺点

- 插入和删除操作需要移动大量元素。
- 线性表变化较大时，难以确定存储空间的容量。
- 存储分配需要一整段连续的存储空间，不够灵活。

> 注意： 动态分配的数组还还是**属于**顺序存储结构，动态分配并不是链式存储，同样是顺序存储，其物理结构没有发送变化，依然是随机存取方式，只是分配的空间大小可以在运行时决定。

动态分配形式 `int * p = (int *)malloc(size of(int)*10)`

C的初始动态分配语句为：

```
L.data=(ElemType*)malloc(sizeof(ElemType)*InitSize);
```

C++的初始动态分配语句为：

```
L.data=new ElemType[InitSize];
```

> 有序： 不一定是按从小到大或从大到小