嵌入式C语言之-

数组越界的危害

讲师：叶大鹏

程序能正确执行吗?

```
int main(void)
{
        volatile int32_t sum;
        volatile int32_t x = 100, y = 200;
        volatile int32_t nums[4];

        for (uint8_t i = 0; i <= 4; i++)
        {
                nums[i] = i;
        }
        sum = x + y;
        printf("sum = %d.\n", sum);
        return 0;
}
```

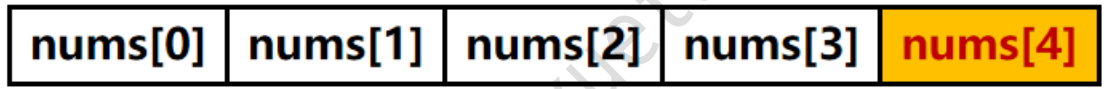
程序能正确执行吗?

```
int main(void)
{
        volatile int32_t sum;
        volatile int32_t x = 100, y = 200;
        volatile int32_t nums[4];

        for (uint8_t i = 0; i <= 4; i++)
        {
                nums[i] = i;
        }
        sum = x + y;
        printf("sum = %d.\n", sum);
        return 0;
}
```

数组越界

```
int32_t nums[4];
nums[4] = 0;
```

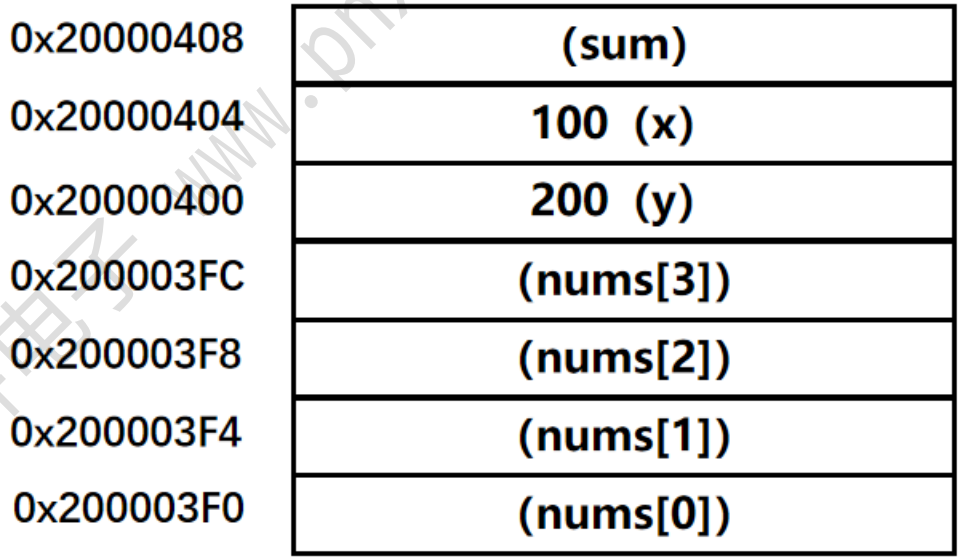


- 数组越界是指数组下标取值超过了数组元素数量大小，导致对数组元素的访问出现在数组的范围之外，这类错误是 C 语言程序中最常见的错误之一；
- 我们使用nums[4]这个元素相当于我们在非法使用权限范围之外的存储空间，那么这块存储空间原本存储的数据是什么我们不知道，可能有用，也可能没用；在给nums[4]这个元素赋值时，会对这块存储区里的数据进行覆盖，万一这里存储的是有用的数据，就会出现BUG。
- C语言赋予了指针和数组极大的灵活性，即使越界也不会编译报错，所以检验数组的边界是程序员的职责，需要时刻保持敏感。

问题分析

- 依赖局部变量生命周期、栈工作原理、数组内存分布的知识点。

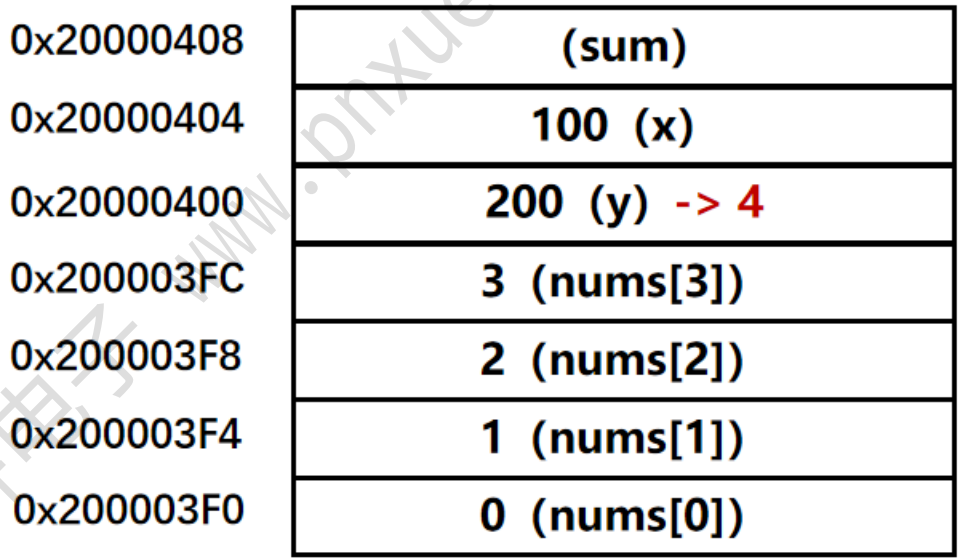
```
volatile int32_t sum;
volatile int32_t x = 100, y = 200;
volatile int32_t nums[4];
```



main函数的栈空间

问题分析

```
for (uint8_t i = 0; i <= 4; i++)
{
        nums[i] = i;
}
```

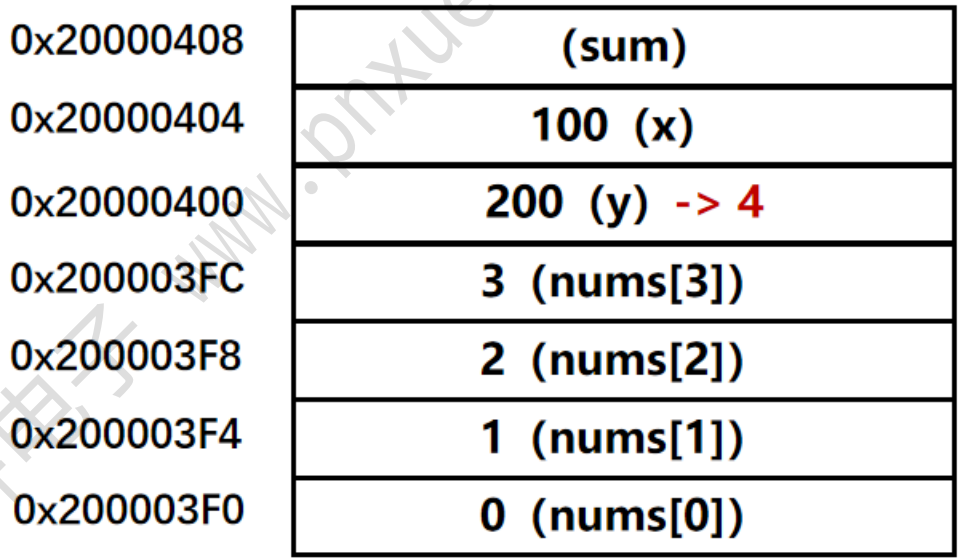


main函数的栈空间

- 变量y在内存中的数据被篡改了，由200变为了4。

问题分析

```
sum = x + y;
printf("sum = %d.\n", sum);
```



main函数的栈空间

- 最终运算的结果是100 + 4，而不是100 + 200。

THANK YOU!