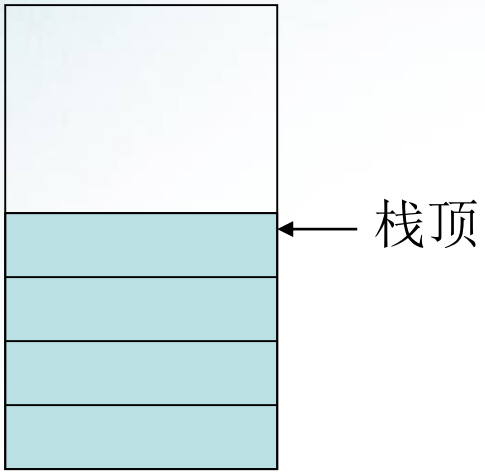
顺序栈

顺序栈



- 如何存储栈中的元素？
  - 顺序栈需要连续的一块内存空间存储元素——数组
  - 数组的**maxsize**是固定的。某一时刻，仅占用数组的一部分
- 栈中元素的个数是多少？
  - 计数器
- 注意：顺序栈和数组的差别

顺序栈的定义

```
const int maxstack = 10; // small value for testing
class Stack {
public:
    Stack( );
    bool empty( ) const;
    Error_code pop( );
    Error_code top(Stack_entry &item) const;
    Error_code push(const Stack_entry &item);
private:
    int count;
    Stack_entry entry[maxstack];
};
```

顺序栈

- 栈底相对固定，通常选择0号单元格作为栈底
- 栈顶与计数器之间的关系
- **两个概念：**
  - **上溢**
  - **下溢**

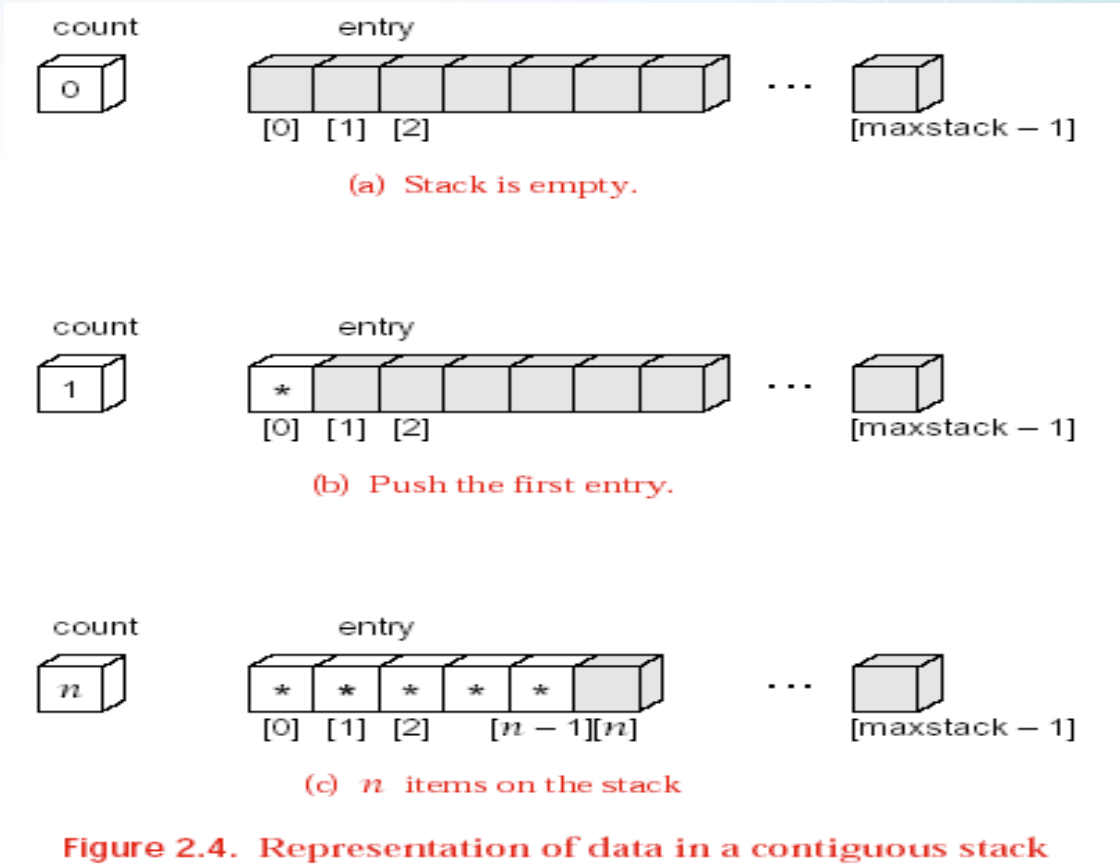


Figure 2.4. Representation of data in a contiguous stack

顺序栈中核心方法的实现

```
Error_code Stack :: push(const Stack_entry &item)
/* Pre: None.
Post: If the Stack is not full, item is added to the top of the Stack. If the
Stack is full, an Error_code of overflow is returned and the Stack is left
unchanged.
*/
{
    Error_code outcome = success;
    if (count >= maxstack)
        outcome = overflow;
    else
        entry[count++] = item;
    return outcome;
}
```

顺序栈中核心方法的实现

```
Error_code Stack :: pop( )
/* Pre: None.
Post: If the Stack is not empty, the top of the Stack is removed. If the
Stack is empty, an Error_code of underflow is returned. */
{
    Error_code outcome = success;
    if (count == 0)
        outcome = underflow;
    else --count;
        return outcome;
}
```

顺序栈中核心方法的实现

```
Error_code Stack :: top(Stack_entry &item) const
/* Pre: None.
Post: If the Stack is not empty, the top of the Stack is returned in item.
If the Stack is empty an Error_code of underflow is returned. */
{
    Error_code outcome = success;
    if (count == 0)
        outcome = underflow;
    else
        item = entry[count - 1];
    return outcome;
}
```

顺序栈中核心方法的实现

```
bool Stack :: empty( ) const
/* Pre: None.
Post: If the Stack is empty, true is returned. Otherwise false is
returned. */
{
    bool outcome = true;
    if (count > 0) outcome = false;
    return outcome;
}
```

```
Stack :: Stack( )
/* Pre: None.
Post: The stack is initialized to be empty. */
{
    count = 0;
}
```