栈

学习目标

熟知栈的定义

掌握栈的操作特性

了解栈的抽象数据类型定义

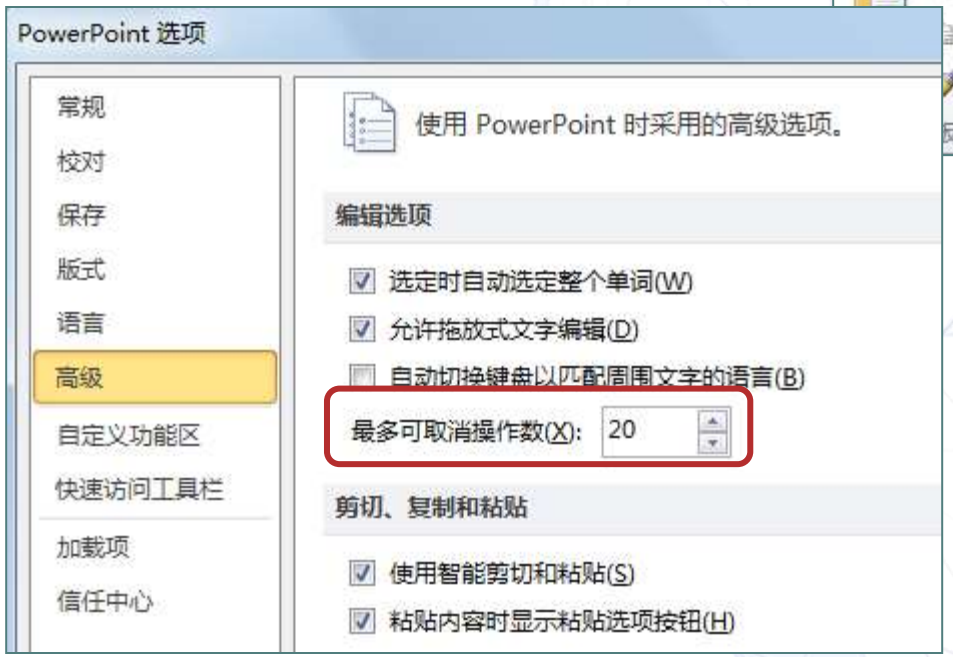
掌握栈的存储结构(顺序栈和链式栈)

熟练进行栈的实现

随处可见的栈结构

Office的撤销机制

- 人生无法后悔，所以且行且珍惜！
- 计算机后悔很容易，所以大胆往前走！



括号匹配问题

【问题】 判断给定表达式中所含括号是否正确配对。

【想法】 配对原则：右括号与其前面最近的尚未配对的左括号相配对。
顺序扫描表达式，将右括号与已经扫描过的最后一个尚未配对的左括号进行配对。

$5 + ((3 + 2) \times 8 - 7) \div 3$

$5 + ((3 + 2) \times 8 - 7)) \div 3$

如何保存已经扫描过的尚未配对的左括号，并对其实施配对操作？

用栈保存，扫描到左括号进栈，扫描到右括号出栈

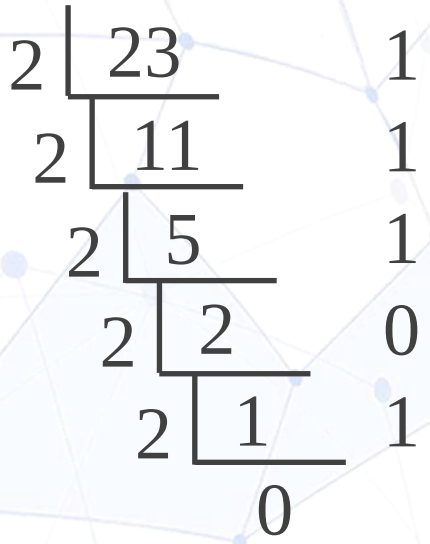
在实际问题的处理过程中，有些数据具有后到先处理的特点

进制转换问题

【问题】 将十进制数转换为二进制数。

【想法】 转换规则：除基取余，逆序排列。

$$(23)_{10} = (10111)_2$$



如何保存得到的余数，使之能够逆序输出？

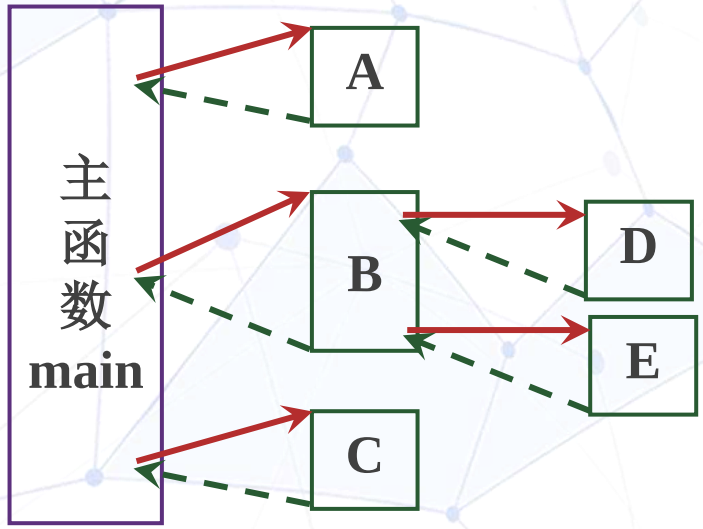
用栈保存，从最后进栈的元素开始输出

在实际问题的处理过程中，有些数据具有后到先处理的特点

函数嵌套调用

【问题】嵌套调用：在函数的执行过程中调用其他函数，返回到哪里？

【想法】为保证函数嵌套调用的正确执行，返回到调用位置。



如何保存调用位置？

用栈保存，返回最后进栈的位置

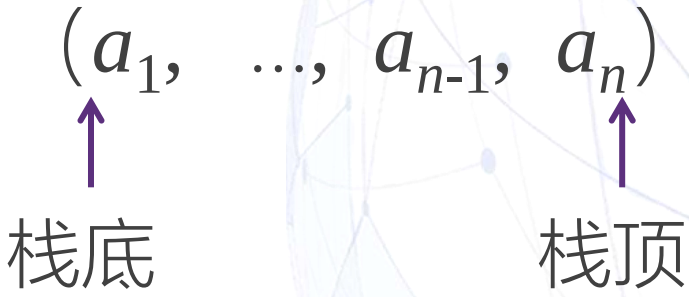
在实际问题的处理过程中，有些数据具有后到先处理的特点

关于栈结构

- 什么是栈？在逻辑上有什么特点？在操作上有什么特性？
- 如何存储栈结构？
- 在不同的存储结构上，如何实现插入、删除、查找等基本操作？
- 在不同的存储结构上，基本操作的时空性能如何？

栈的定义

- 栈：限定仅在一端进行插入和删除操作的线性表

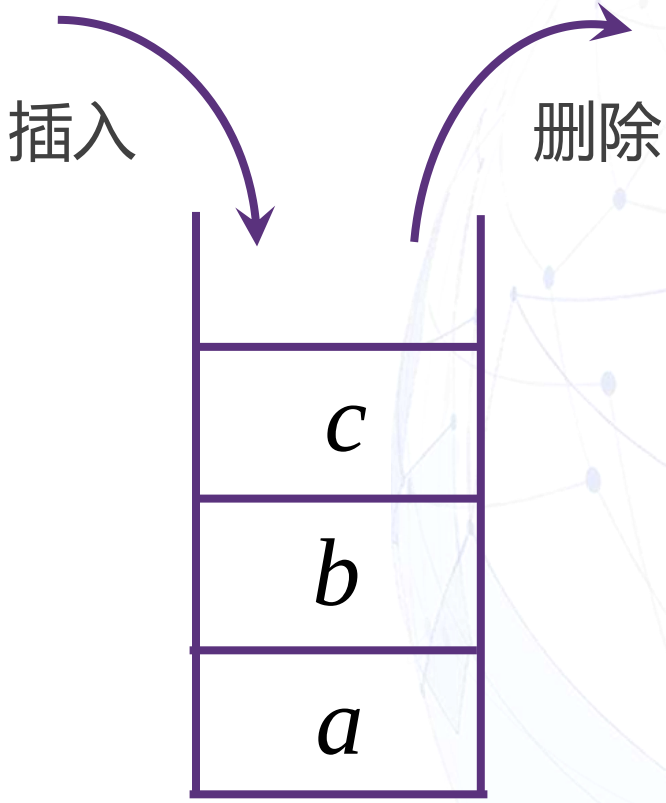


线性表
- 插入位置：$1 \sim n+1$
- 删除位置：$1 \sim n$

栈
- 插入位置：$n+1$
- 删除位置：n

- 栈顶（top）：允许插入和删除的一端称为栈顶
  栈底（bottom）：另一端称为栈底

栈的操作特性



插入：入栈、进栈、压栈

删除：出栈、弹栈

空栈：不含任何数据元素的栈

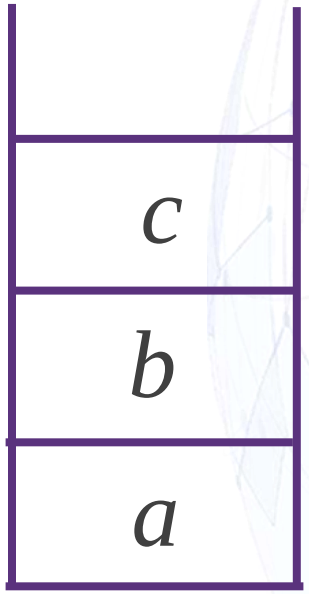
条件判断

此时执行出栈操作，哪个元素可以出栈呢?

栈的操作特性：后进先出（**L**ast **I**n **F**irst **O**ut，LIFO）
先进后出（**F**irst **I**n **L**ast **O**ut，FILO）

栈的操作特性

例：有三个元素按a、b、c的次序依次进栈，且每个元素只允许进一次栈，则可能的出栈序列有多少种？



情况一

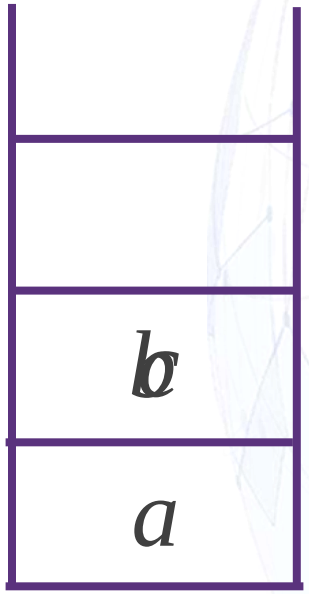
出栈：$c\ b\ a$

栈只是对插入和删除操作的位置进行了限制
并没有限定插入和删除操作进行的时间

即不排除后者未进栈之前，先入栈者先出栈的情况

栈的操作特性

例：有三个元素按a、b、c的次序依次进栈，且每个元素只允许进一次栈，则可能的出栈序列有多少种？

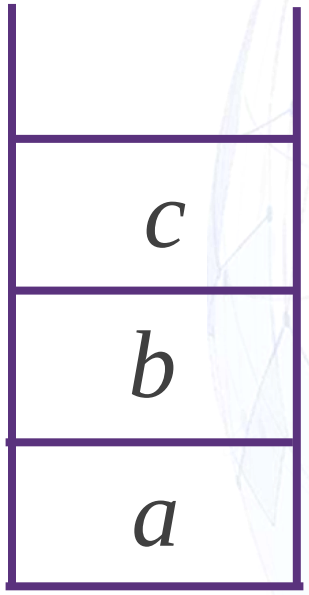


情况二

出栈：$b\ c\ a$

栈的操作特性

例：有三个元素按a、b、c的次序依次进栈，且每个元素只允许进一次栈，则可能的出栈序列有多少种？

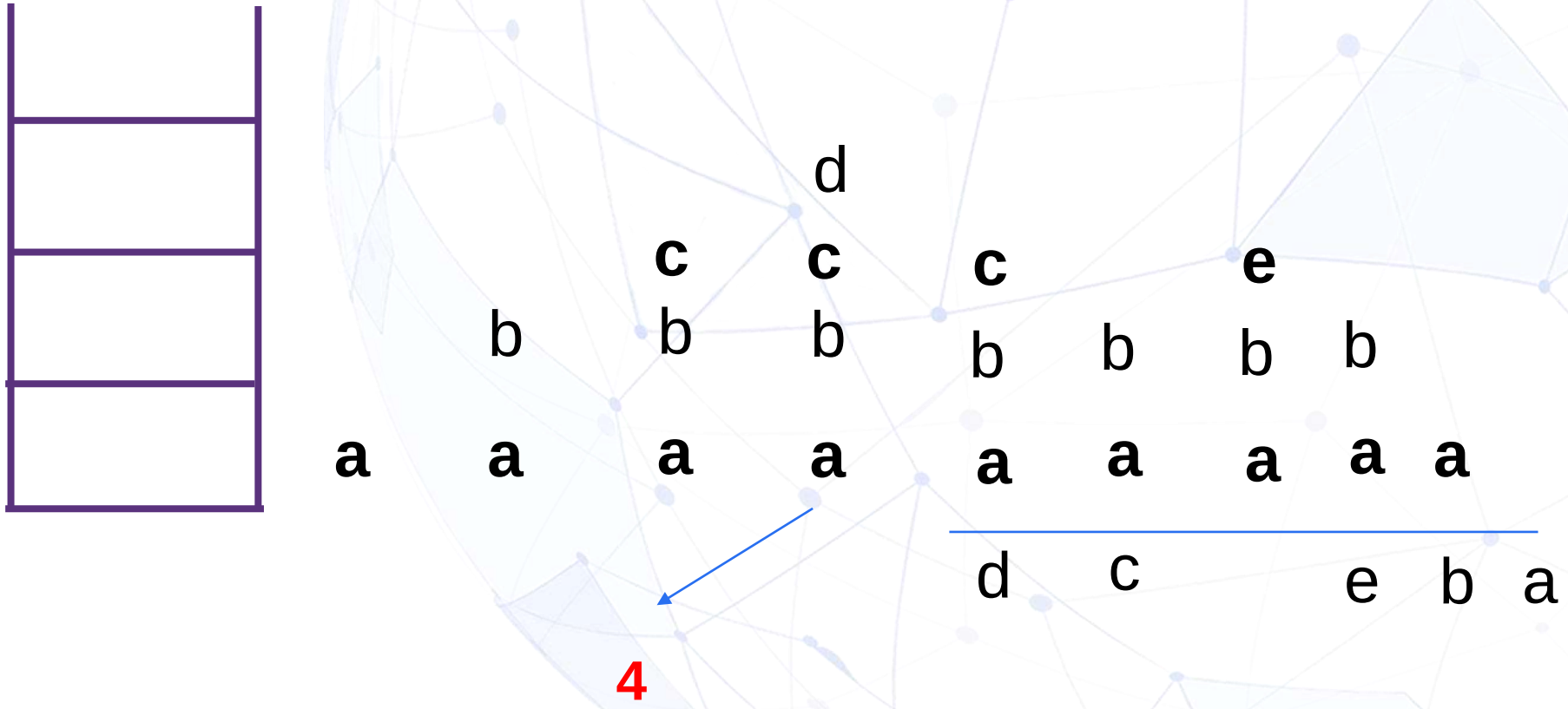


能否得到如下出栈序列？

出栈：$c\ a\ b$

abc　acb　bac　bca　cba

例：假设有 5 个元素abcde顺序进栈（进栈过程中可以出栈），出栈序列为dceba，则该栈容量必定不小于多少？



栈的抽象数据类型定义

ADT Stack

DataModel

栈中元素具有相同类型及后进先出特性，相邻元素具有前驱和后继关系

Operation

InitStack：栈的初始化

DestroyStack：栈的销毁

Push：入栈

Pop：出栈

GetTop：取栈顶元素

Empty：判空

endADT

相对于其他数据结构，栈的基本操作是确定的

栈的抽象数据类型定义

InitStack

输入：无

功能：栈的初始化，初始化一个空栈

输出：无

DestroyStack

输入：无

功能：销毁栈，释放栈所占用的存储空间

输出：无

Push

输入：元素值 x

功能：在栈顶插入一个元素 x

输出：如果插入成功，栈顶增加了一个元素，否则返回失败信息

Push操作需要指明插入位置吗？

栈的抽象数据类型定义

Pop
- 输入：无
- 功能：删除栈顶元素
- 输出：如果删除成功，返回被删元素值；否则返回失败信息

GetTop
- 输入：无
- 功能：读取当前的栈顶元素
- 输出：若栈不空，返回当前的栈顶元素值；否则返回失败信息

Empty
- 输入：无
- 功能：判断栈是否为空
- 输出：如果栈为空，返回 1；否则，返回 0

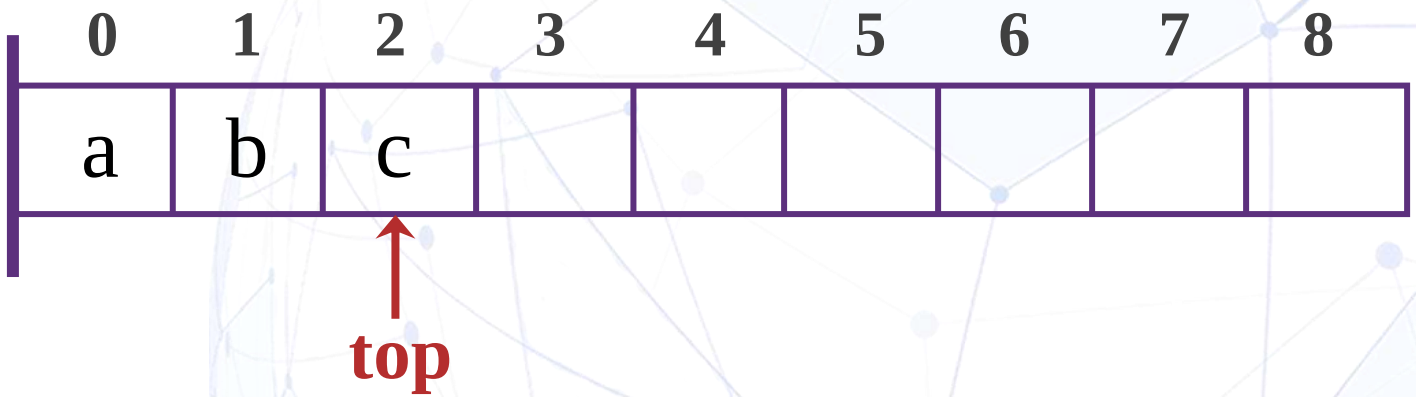
栈的存储结构(顺序栈)

- 顺序栈的存储结构定义
- 顺序栈的实现——入栈
- 顺序栈的实现——出栈
- 顺序栈的实现——取栈顶元素
- 顺序栈的实现——判空

顺序栈的存储结构定义

顺序栈：栈的顺序存储结构

什么是顺序存储？



如何改造数组实现栈的顺序存储呢？

如何表示栈底：用数组的一端作为栈底

如何表示栈顶：设变量top存储栈顶元素所在的下标

顺序栈的类定义

栈的抽象数据类型定义?

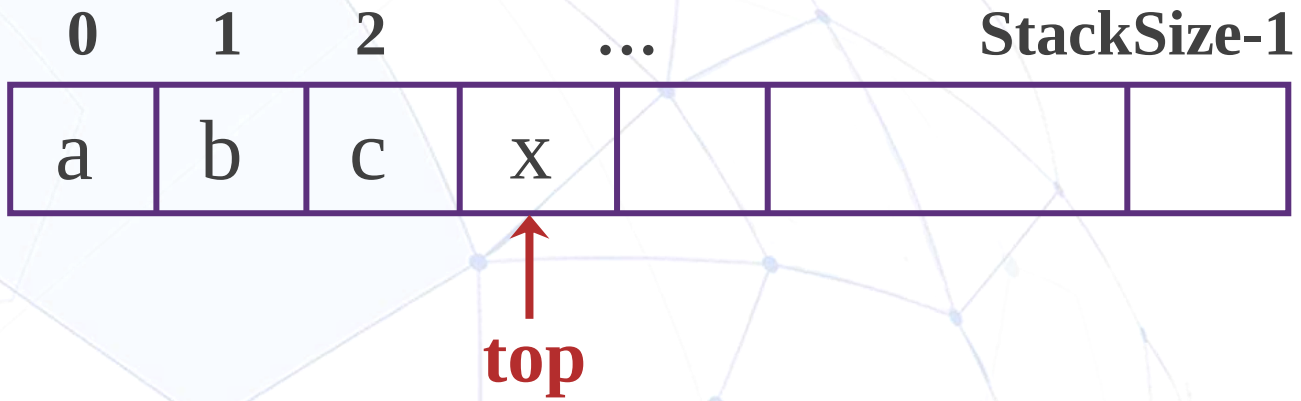
InitStack：栈的初始化
DestroyStack：栈的销毁
Push：入栈
Pop：出栈
GetTop：取栈顶元素
Empty：判空

```
const int StackSize = 10;
template <typename DataType>
class SeqStack
{
public:
    SeqStack( );
    ~SeqStack( );
    void Push( DataType x );
    DataType Pop( );
    DataType GetTop( );
    int Empty( );
private:
    DataType data[StackSize];
    int top;
};
```

栈的初始化

```
SeqStack<DataType> :: SeqStack(){
        top=-1;
}
```

顺序栈的实现——入栈

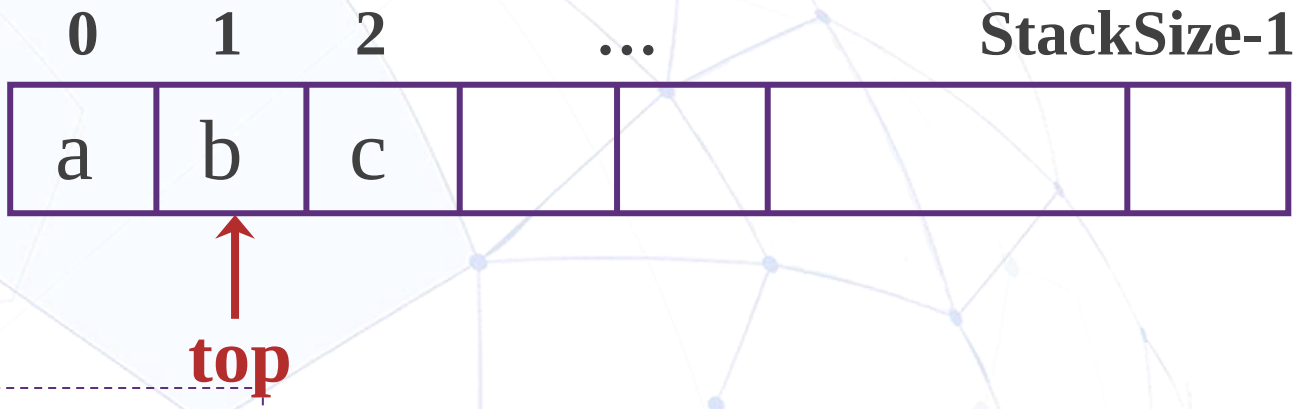


栈满：top=StackSize−1

```
template <typename DataType>
void SeqStack<DataType> :: Push(DataType x)
{
    if (top == StackSize - 1) throw "上溢";
    data[++top] = x;
}
```

- 什么情况下无法入栈?
- 时间复杂度?

顺序栈的实现——出栈



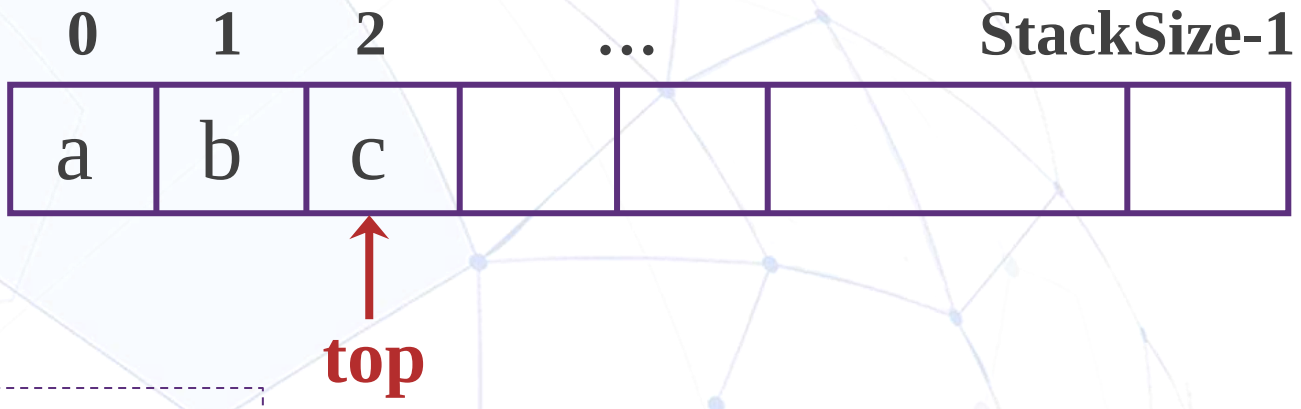

```
template <typename DataType>
DataType SeqStack<DataType> :: Pop( )
{
    if (top == -1) throw "下溢";
    DataType x;
    x = data[top--];
    return x;
}
```

栈空：top=−1

什么情况下无法出栈?

取栈顶元素的实现?

顺序栈的实现——取栈顶元素



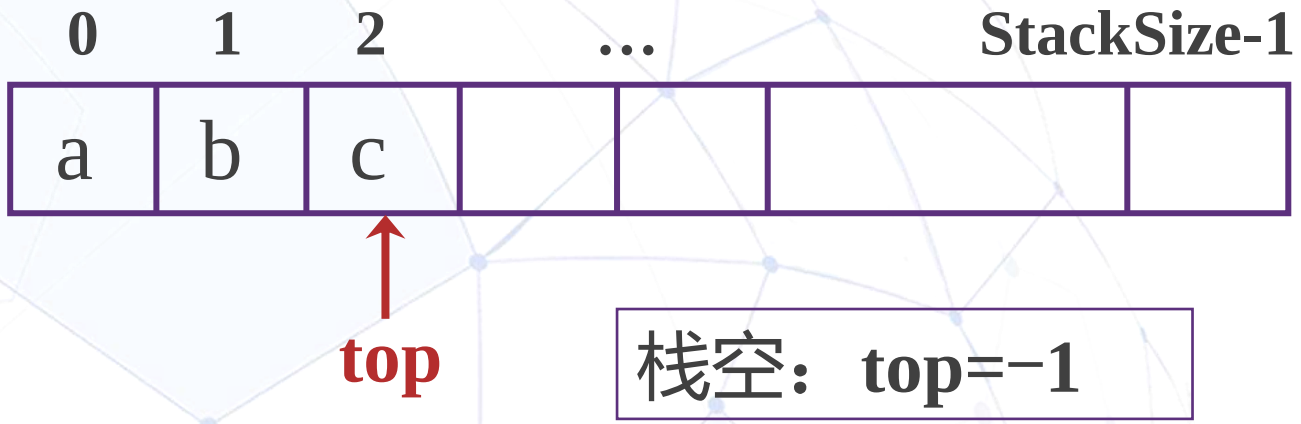

```
template <typename DataType>
DataType SeqStack<DataType> ::  GetTop ( )
{
    if (top == -1) throw "下溢";
    DataType x;
    x = data[top];
    return x;
}
```

什么情况下无法取栈顶?

栈空：**top=−1**

顺序栈的实现——判空

判空的函数原型是什么？



栈空：top=−1

Empty

输入：无

功能：判断栈是否为空

输出：如果栈为空，返回 1；否则，返回 0

```
template <typename DataType>
int  SeqStack<DataType> :: Empty( )
{
      if (top == -1) return 1
      else return 0;
}
```

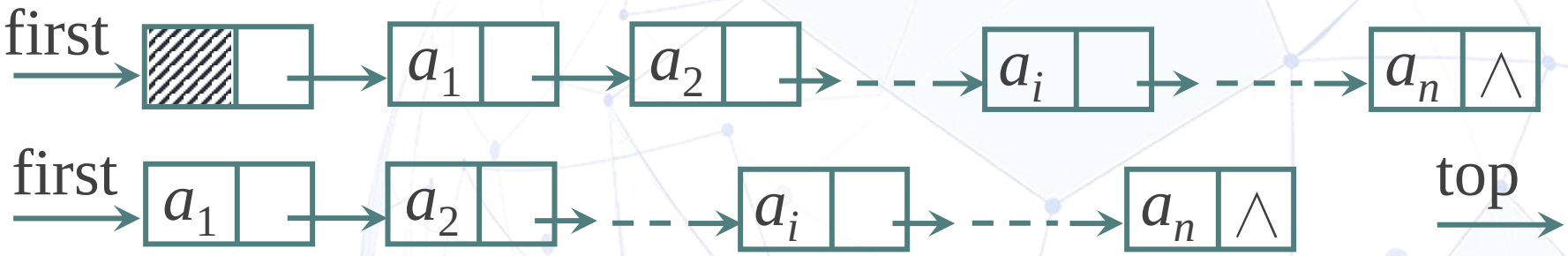
栈的存储结构(链式栈)

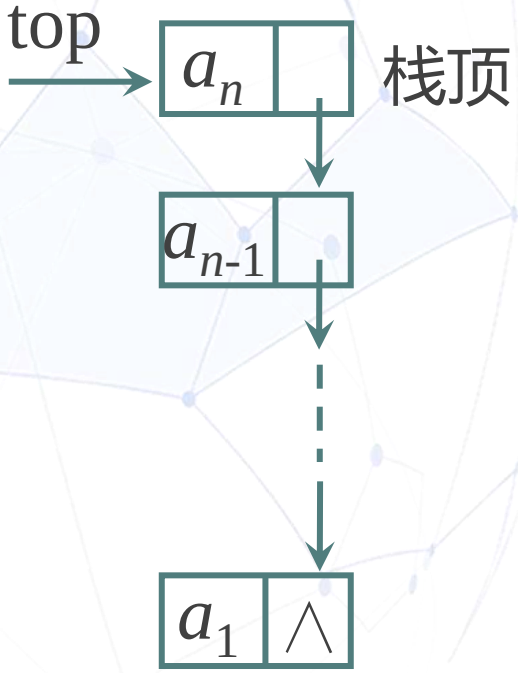
- 链栈的存储结构定义
- 链栈的实现——入栈
- 链栈的实现——出栈
- 链栈的实现——取栈顶元素
- 链栈的实现——判空

链栈的存储结构定义

链栈：栈的链接存储结构

单链表是如何存储线性表的？





P：用链表的哪一端作为栈顶？

A：为方便操作，用链头作为栈顶

P：链栈需要加头结点吗？

P：单链表为什么加头结点？

A：链栈无须加头结点

✓ 指针的方向从栈顶向下链接

链栈的存储结构定义

栈的抽象数据类型定义?

InitStack：栈的初始化
DestroyStack：栈的销毁
Push：入栈
Pop：出栈
GetTop：取栈顶元素
Empty：判空

```
template <typename DataType>
class LinkStack
{
public:
    LinkStack( );
    ~LinkStack( );
    void Push(DataType x);
    DataType Pop( );
    DataType GetTop( );
    int Empty( );
private:
    Node<DataType> *top;
};
```

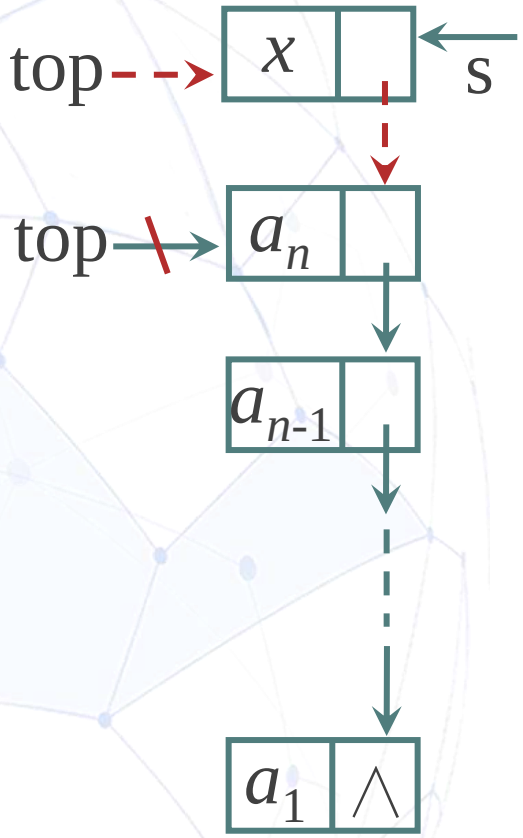
链栈初始化和销毁

```
LinkStack<DataType> :: LinkStack (){
        //链栈不带头结点，直接top置为空
          top=nullptr;
}

LinkStack<DataType> ::~ LinkStack (){
      //析构函数，删除链栈，要释放栈中每个结点，跟单链表析构函数类似
          Node<DataType> *p=top;
          while（top!=nullptr）
          {
                top=top->next;
                delete p;
                p=top;
          }
}
```

链栈的实现——入栈

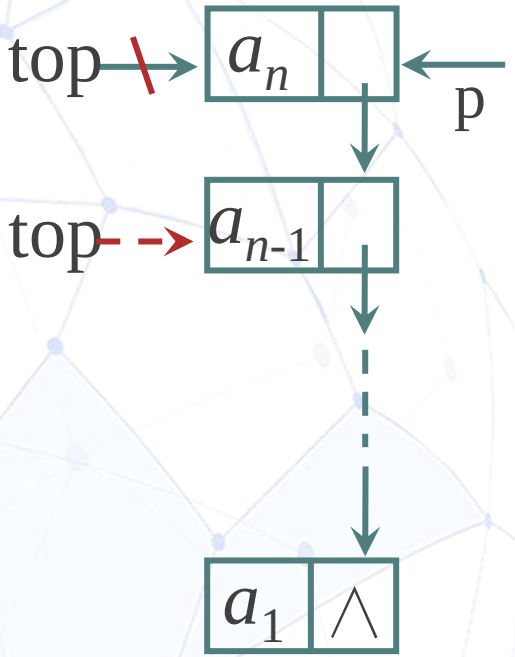
```
template <typename DataType>
void LinkStack<DataType> :: Push(DataType x)
{
    Node<DataType> *s = nullptr;
    s = new Node<DataType>;
    s->data = x;                          //申请结点s数据域为x
    s->next = top; top = s;               //将结点s插在栈顶
}
```



链栈的入栈操作为什么不用判断是否栈满?

链栈的实现——出栈

```
template <typename DataType>
DataType LinkStack<DataType> :: Pop( )
{
    Node<DataType> *p = nullptr;
    DataType x;
    if (top == nullptr) throw "下溢";
    x = top->data;
    p = top;                    //暂存栈顶元素
    top = top->next;            //将栈顶结点摘链
    delete p;
    return x;
}
```



空栈　top = nullptr

什么情况下无法删除?

取栈顶元素的实现?

顺序栈VS链栈

时间复杂度

链栈和顺序栈的基本操作时间复杂度均为O（1）

空间复杂度

顺序栈要确定一个固定的长度，所以有存储个数的限制和浪费空间的问题；

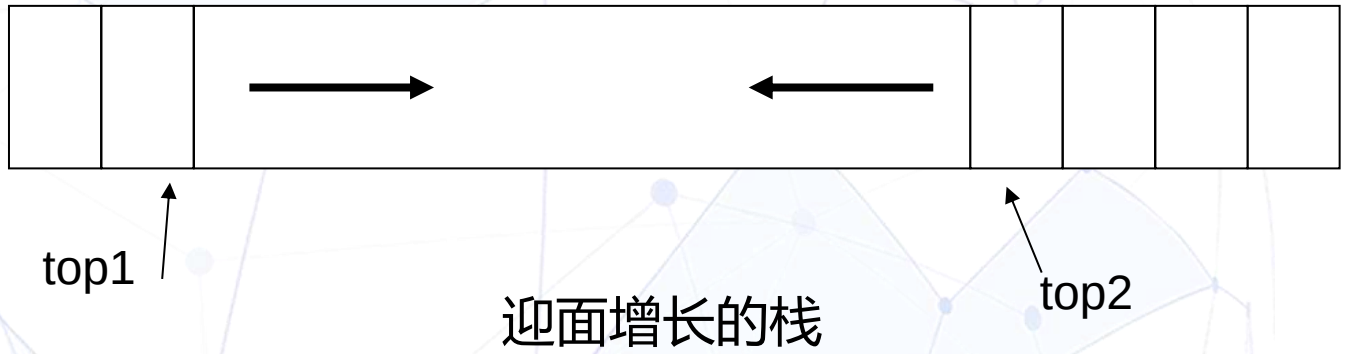
链栈原理上没有栈满问题，除非内存没有空间，但是每个元素需要带指针域，增加了结构性开销。

栈的变种

两个独立的栈

- ✓ 底部相连：双栈
- ✓ 迎面增长

- ✓ 各自适用的场景？



迎面增长的栈

底部相连的栈

课堂小结

栈的定义和逻辑结构

栈的实现(顺序栈和链式栈)