指针

POINTER

- 今天我们来谈指针，指针是C++的特色，不得不品尝。

- 今天我们来谈指针，指针是C++的特色，不得不品尝。
- 不妨先回顾一下什么是指针

什么是指针?

简单地说

指针就是一个保存内存单元首地址的变量

(一个无符号整数)

总结一下

指针是个变量

指针是变量

做个小实验

```
int   b = 0;
int* pb = &b;
printf("%0xu",pb);
```

输出什么?

每当调用一个变量的时候，程序会：

1. 找到该内存地址
2. 如果有偏移量，那么根据偏移量在内存上进行移动
3. 读取对应内存地址的值

每当通过指针调用一个变量的时候：

1. 找到保存指针的内存地址
2. 如果有偏移量，根据偏移量在内存上进行移动
3. 读取对应内存地址的值
4. 根据刚刚读出的值，再找到对应的内存地址
5. 如果有偏移量，根据偏移量在内存上进行移动
6. 读取对应内存地址的值

多重指针以及

指向数组的指针

```
int p        = 0;
int *pp      = &p;
int **ppp    = &pp;
int ***pppp = &ppp;
```

```
int p        = 0;
int *pp      = &p;
int **ppp    = &pp;
int ***pppp = &ppp;
```

- 套娃

```
int p        = 0;
int *pp      = &p;
int **ppp    = &pp;
int ***pppp = &ppp;
```

- 套娃
- 多重指针，简单地说就是指向前一级指针的指针

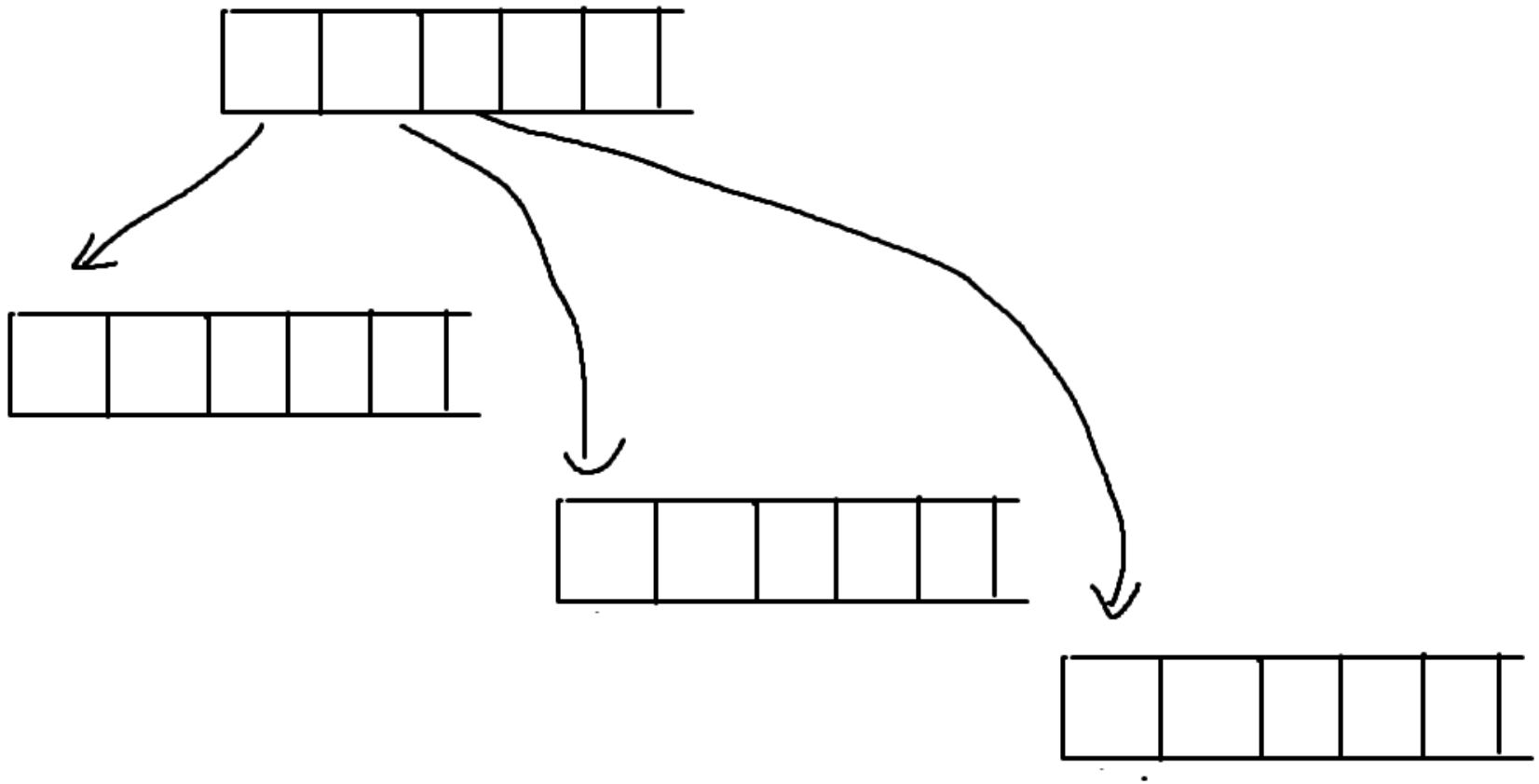
用多重指针实现动态多维数组

```
int **array = new int*[100];
for(int i=0;i<100;i++){
    array[i] = new int[100];
}
```

注：下标运算符优先级为2，结合性为左结合

这种方法创建出来的数组是什么样子的?

多重指针数组



指向数组的指针

```
int p[100];//错误示范
int (*pp)[]= p;
```

```
int p[100][100];
int (*pp)[100]=p;
```

指针步长：对指针进行+1操作时，指针指向的内存地址偏移量

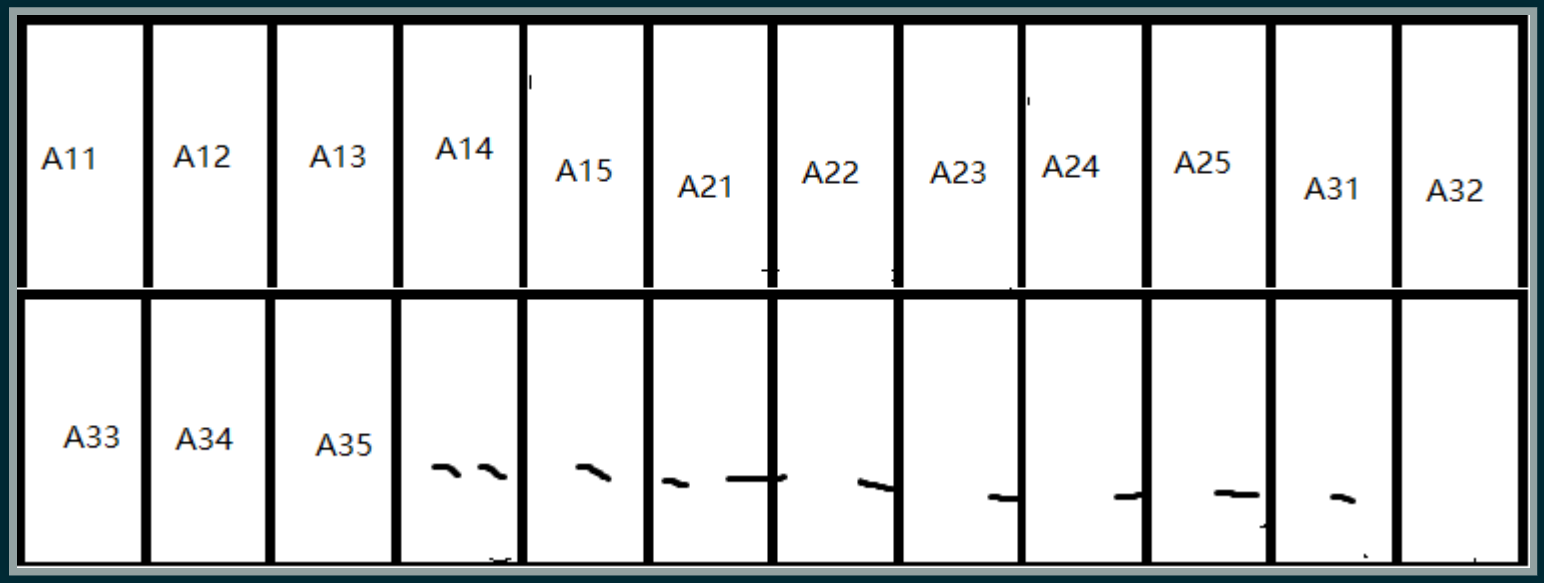
指针步长取决于它指向的对象有多大

二维数组的数组名会被隐式转换成指向一维数组的指针

```
int a[100][100];//a[0]是一个int *类型，a是一个int (*)[100]类型
```

多维数组的存储方式

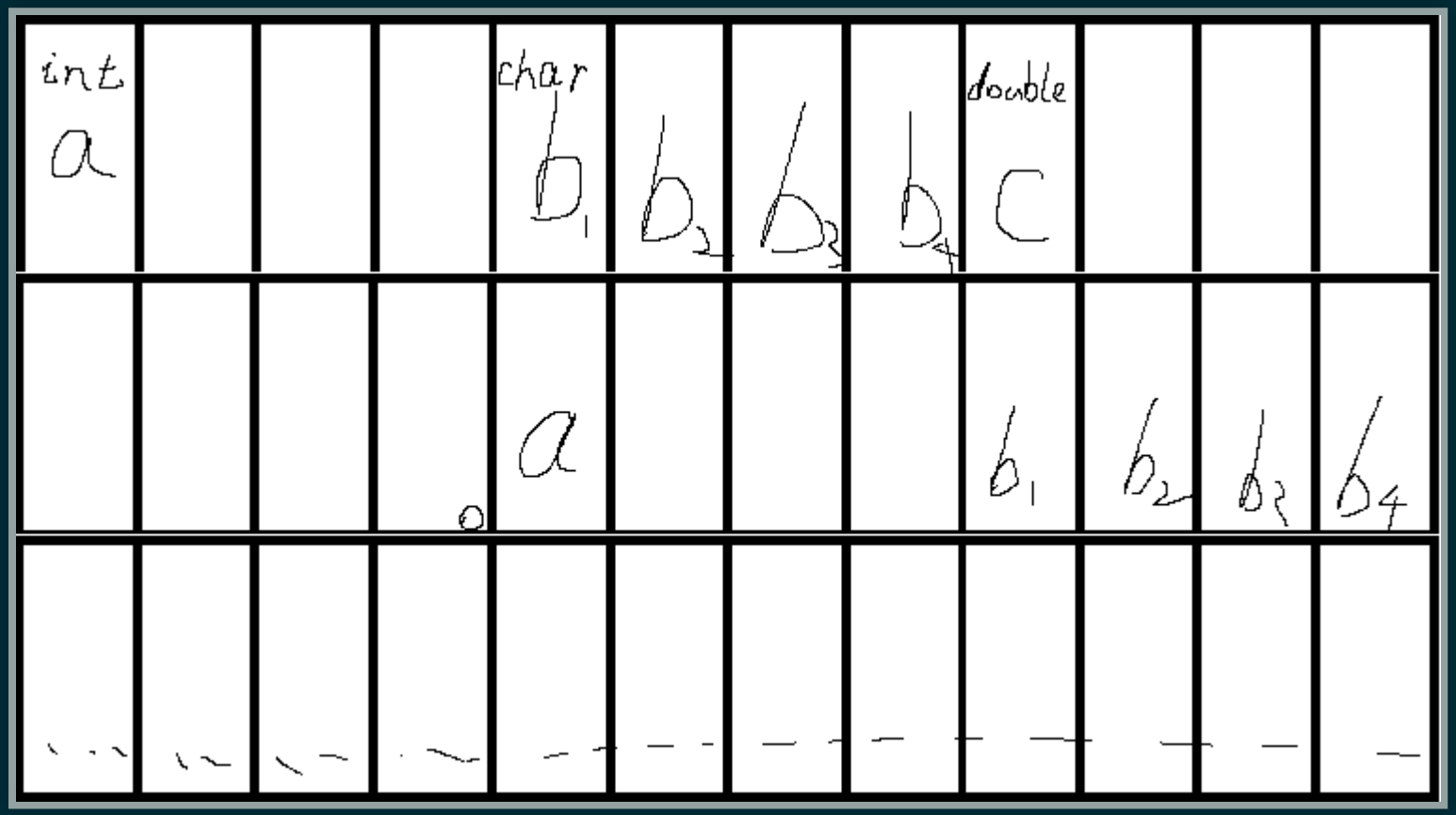
对于二维数组A，它在内存中是连续排放的



E1[E2]等价于(E1+E2)，而E1[E2][E3]*
由于左结合性等价于(E1[E2])[E3]

结构体的内存分配

一图流



面向对象中的指针

基类指针指向派生类对象

基类指针可以指向派生类对象

派生类指针不能指向基类对象

```
class A{
public:
    void print(){printf("A")};
};
class B : public A{
public:
    void print(){printf("B")};
};

A a;
a.print();//输出A
B b;
b.print();//输出B

A *p = b;
p->print();//输出A
```

```cpp
class A{
public:
    void print(){printf("A")};
};
class B : public A{
public:
    void print(){printf("B")};
};

A a;
a.print();//输出A
B b;
b.print();//输出B

A *p = b;
p->print();//输出A
```

- 看似对于每个对象实现了各自的方法，但这并不是真正意义上的多态

面向对象中的指针

多态的实现

```
class A{
public:
    virtual void print(){printf("A")};
};
class B : public A{
public:
    void print(){printf("B")};//virtual可加可不加
}
B b;
A *p = b;
p->print();//输出B
```

```
class A{
public:
    virtual void print(){printf("A")};
};
class B : public A{
public:
    void print(){printf("B")};//virtual可加可不加
}
B b;
A *p = b;
p->print();//输出B
```

- 在调用函数时会根据指向的对象改变调用的方法

什么时候可能会用到?

比如，在STL容器里存放对象的时候，如果只存放基类就会损失掉派生类额外的信息，所以转而保存基类指针

由于基类指针无法直接访问派生类对象的额外信息，所以在使用的时候还要进行强制类型转换

安利

额外内容

STL库中的智能指针