

# 计算机程序设计基础


Programming Fundamentals

韩文弢

清华大学计算机系



# 第七章 结构体

- 
- 1 结构体的定义和使用
  - 2 结构体数组与指针
  - 3 结构体作为函数参数

# 7.1.1 引言



1.	2.	3.	4.	5.	6.	7.	8.
							
Cao Cao	Sun Jian	Liu Bei	Yuan Shao	Yuan Shu	Liu Biao	Dong Zhuo	Liu Yan
Body	(Basic value: 85)	--	84	--	Hit Space Bar		
Intell.	(Basic value: 95)	--	91	--	Hit Space Bar		
Power	(Basic value: 63)	--	62	--	Hit Space Bar		
Charisma	(Basic value: 99)	--	100	--	Hit Space Bar		
Luck	(Basic value: 71)	--	66	--	Hit Space Bar		



## 刘备的人马

姓名	体力	智力	武力	魅力	运气
刘备	84	91	62	100	66
诸葛亮	80	100	70	95	80
关羽	95	88	99	85	80
张飞	95	85	99	70	70
赵云	100	90	95	90	80
...					

## 新君主的人马

姓名	体力	智力	武力	魅力	运气
新君主	84	91	90	90	85
随从	80	80	80	80	80

如何组织这些数据？

1. 单个人物
2. 所有人物

## 【结构体】

由一个或多个变量（类型可以相同，也可以不同，称为称为字段或成员变量）所组成的一个组合项。

## 7.1.2 结构体的定义

定义一个结构体(struct)的步骤通常是：

1. 定义一个新的结构体类型，并指明它内部的各个成员变量；
2. 使用该类型来定义相应的结构体变量。

结构体类型的定义只在程序的开头出现一次，而结构体变量的声明可以根据需要在程序中出现多次。

## 定义一个结构体类型

```
struct <结构体标记>
{
    类型名1  成员变量1;
    .....
    类型名n  成员变量n;
};
```

## 定义结构体变量:

```
struct <结构体标记> 变量1, 变量2, ...;
```



```
struct General
```

```
{
```

```
    char Name[20];           // 姓名
```

```
    int Body;                // 体力
```

```
    int Intelligence;        // 智力
```

```
    int Power;               // 武力
```

```
    int Charisma;            // 魅力
```

```
    int Luck;                // 运气
```

```
};
```

```
struct General LiuBei;
```

```
struct General GuanYu, ZhangFei;
```

在定义结构体变量后，将为它分配相应的内存空间。

**X**

```
struct student
{
    char ID[7];
    char name[20];
    char gender;
    int age;
    char phone[9];
    char addr[30];
}x;
```

**sizeof(x) = 72**

ID	<input type="text"/>
name	<input type="text"/>
gender	<input type="text"/>
age	<input type="text"/>
phone	<input type="text"/>
addr	<input type="text"/>

## 7.1.3 结构体变量的使用

对一个结构体变量的最基本的操作就是去访问它的各个成员变量。访问的方式为：

**结构体变量名.成员变量名**

一个成员变量就是一个普通的变量，可以对它进行各种通常的变量操作。

```
struct General x;  
  
strcpy(x.Name, "刘备");  
  
if(x.Charisma > 95)  
    cout << x.Name << "是仁君\n";  
  
if(x.Power > 90 && x.Intelligence < 70)  
    cout << x.Name << "是一介武夫";
```

## Can't And Can

- 不能做什么？

- ☺ 不能直接比较两个结构体变量，  
`struct student x, y; (x > y); (x != y);`
- ☺ 不能用**cin/cout**来输入或输出整个结构体变量。

## Can't And Can

- 能做什么？

- ☺ 能够进行结构体变量的整体赋值，如：  
`struct student x, y; y = x;` （why?）
- ☺ 能够定义一个返回值类型为结构体类型的函数，把被调用函数当中的某个结构体变量的值返回给主调函数。

```
struct student x, y;
```

```
...
```

```
y = x;
```

等价于

```
strcpy(y.ID, x.ID);
```

```
strcpy(y.name, x.name);
```

```
y.gender = x.gender;
```

```
y.age = x.age;
```

```
strcpy(y.phone, x.phone);
```

```
strcpy(y.addr, x.addr);
```

**X**

ID 020449

name Zhang

gender M

age 19

phone 62771100

addr 紫荆6-401




```

struct student
{
    char    ID[7];
    char    name[20];
    char    gender;
    int     age;
    char    phone[9];
    char    addr[30];
}x, y;
y = x;

```

14:	y = x;	
→ 0040FA08	mov	ecx,12h
0040FA0D	mov	esi,offset _x (00427e80)
0040FA12	mov	edi,offset _y (004282c0)
0040FA17	rep movs	dword ptr [edi],dword ptr [esi]
15:	}	

# 第七章 结构体

- 
- 1 结构体的定义和使用
  - 2 结构体数组与指针
  - 3 结构体作为函数参数

## 7.2.1 结构体数组



结构体变量



结构体数组

```
struct student  
stu[1000];
```

stu  
stu[0]

ID	<input type="text"/>
name	<input type="text"/>
gen	<input type="text"/>
age	<input type="text"/>
phone	<input type="text"/>
addr	<input type="text"/>

stu[1]

ID	<input type="text"/>
name	<input type="text"/>
gen	<input type="text"/>
age	<input type="text"/>
phone	<input type="text"/>
addr	<input type="text"/>

• • • • •

## 7.2.2 结构体与指针

新类型：基类型为结构体类型的指针类型。

```
struct StudentRecord  
{  
    char name[10];  
    int id;  
    double score;  
} x, *px;  
px = &x;
```

px



A small rectangular box containing a black dot, representing the memory location of the pointer variable px.

**x**

name



A large rectangular box representing the memory layout of the struct variable x. It contains three fields: 'name' with a long rectangular box, 'id' with a medium rectangular box, and 'score' with a long rectangular box.

id

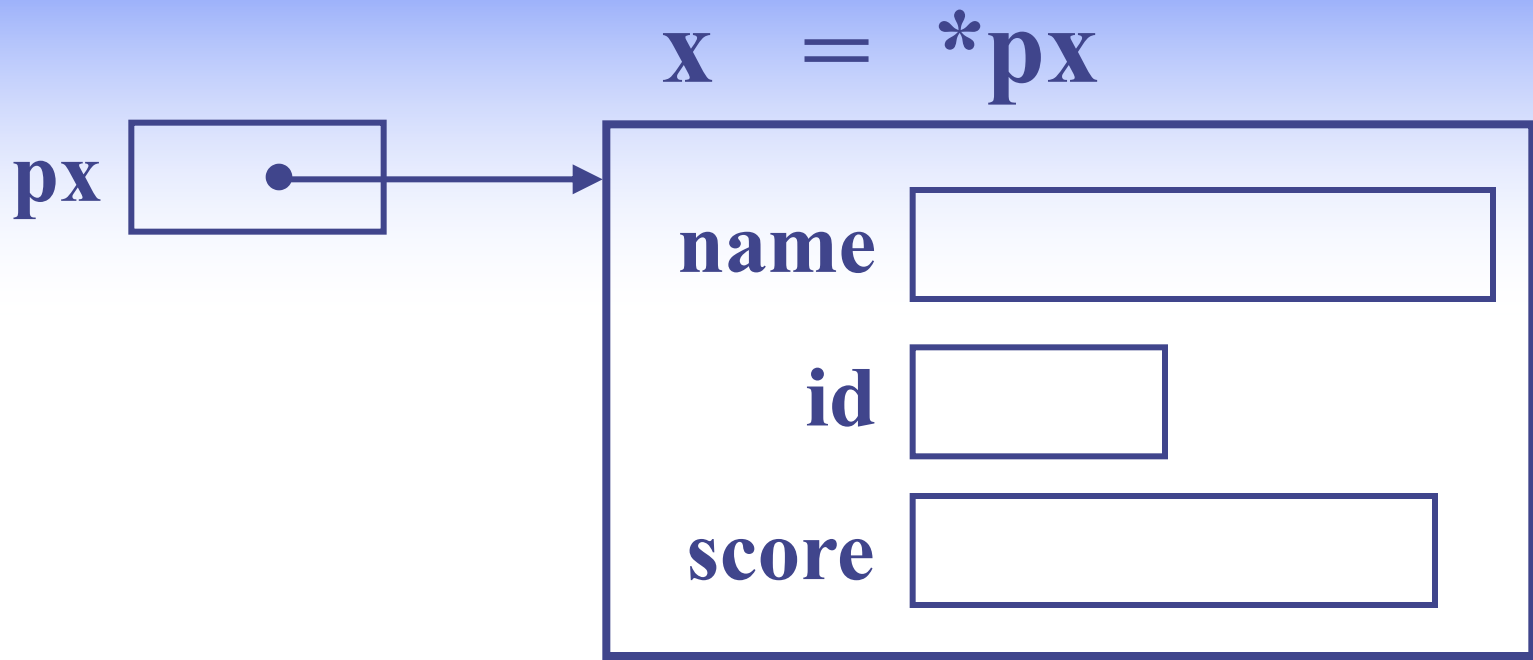


A medium rectangular box representing the memory layout of the 'id' field within the struct variable x.

score

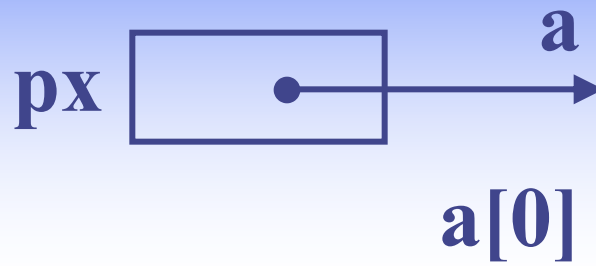


A long rectangular box representing the memory layout of the 'score' field within the struct variable x.



如何来访问x的成员变量？

1. x. 成员变量名，如： **x.name**, **x.id**;
2. (\*px). 成员变量名，如： **(\*px).name**, **(\*px).id**;
3. px->成员变量名， “**->**”称为指向运算符或箭头运算符，如： **px->name**, **px->id**。



```
struct StudentRecord
```

```
    a[10], *px;
```

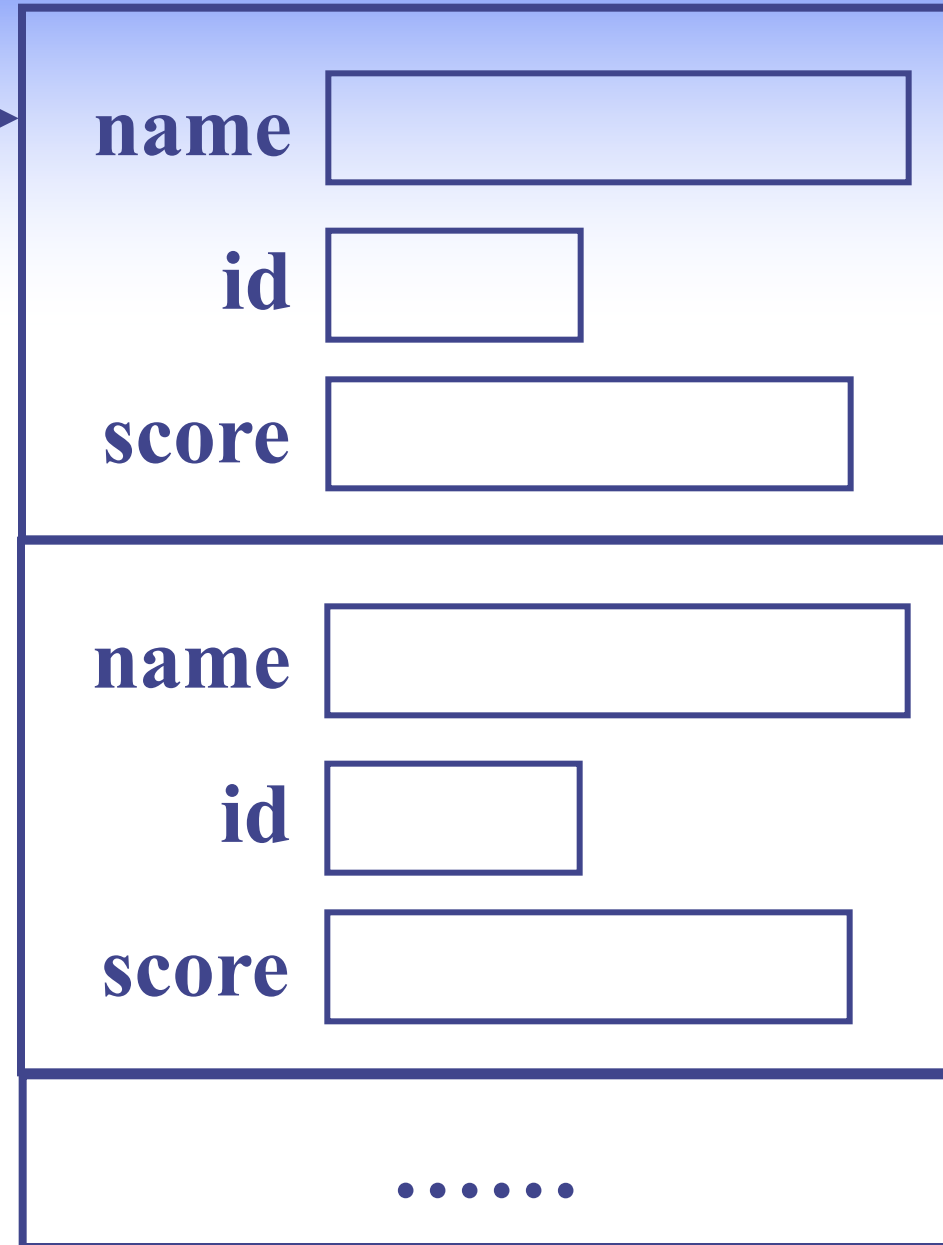
```
    ...
```

```
    px = a;
```


**px** 指向 **a[0]**;  
**px + 1** 指向 **a[1]**;

...

**px + i** 指向 **a[i]**;



# 第七章 结构体

- 
- 1 结构体的定义和使用
  - 2 结构体数组与指针
  - 3 结构体作为函数参数



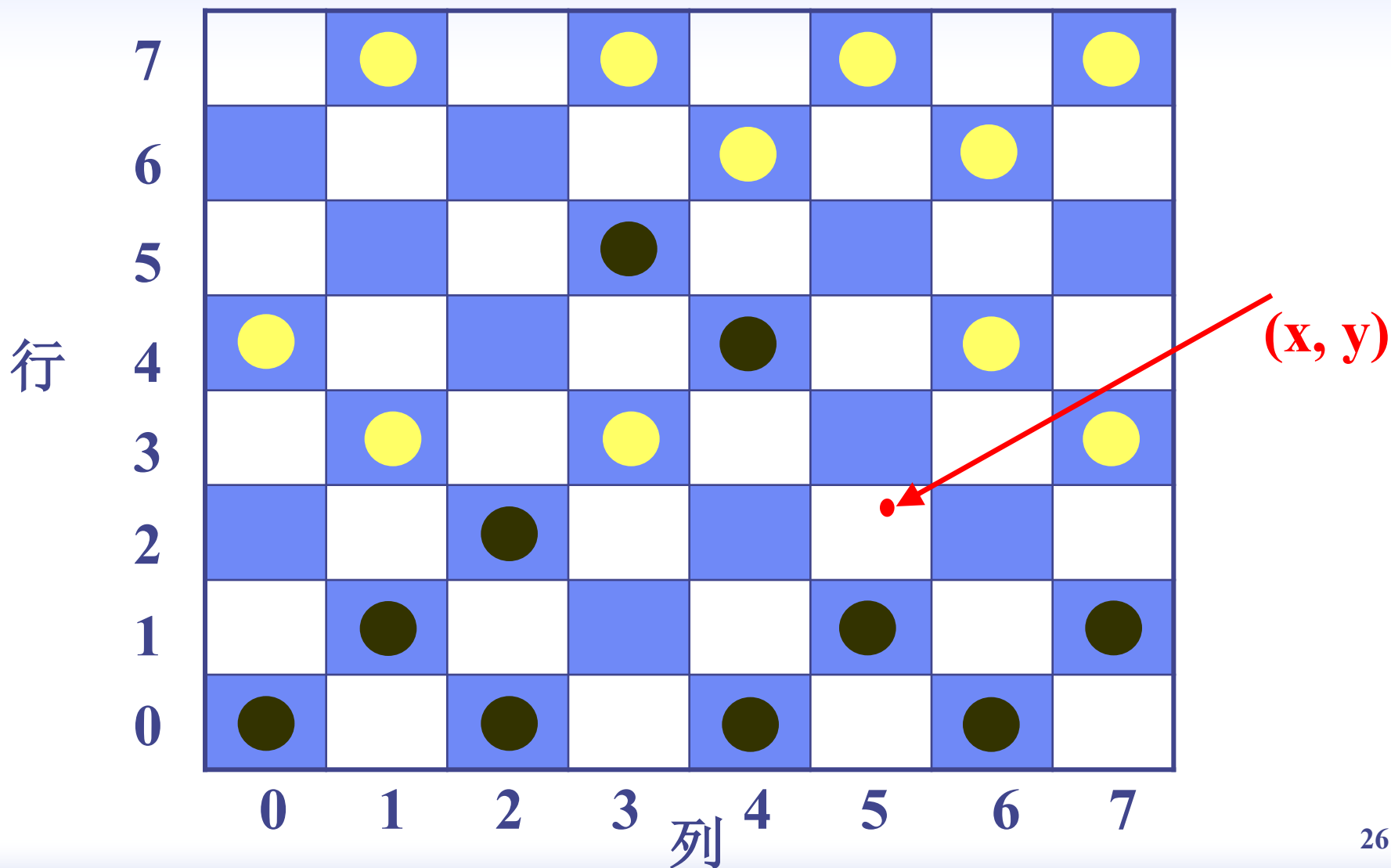
- 函数参数的传递

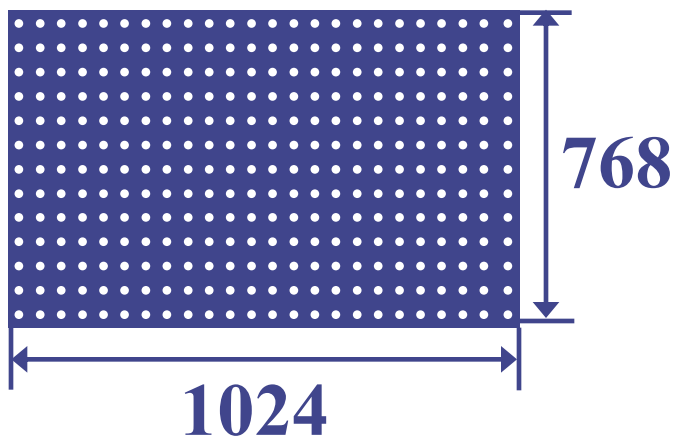
- ☺ 普通变量：传值；

- ☺ 数组：传地址；

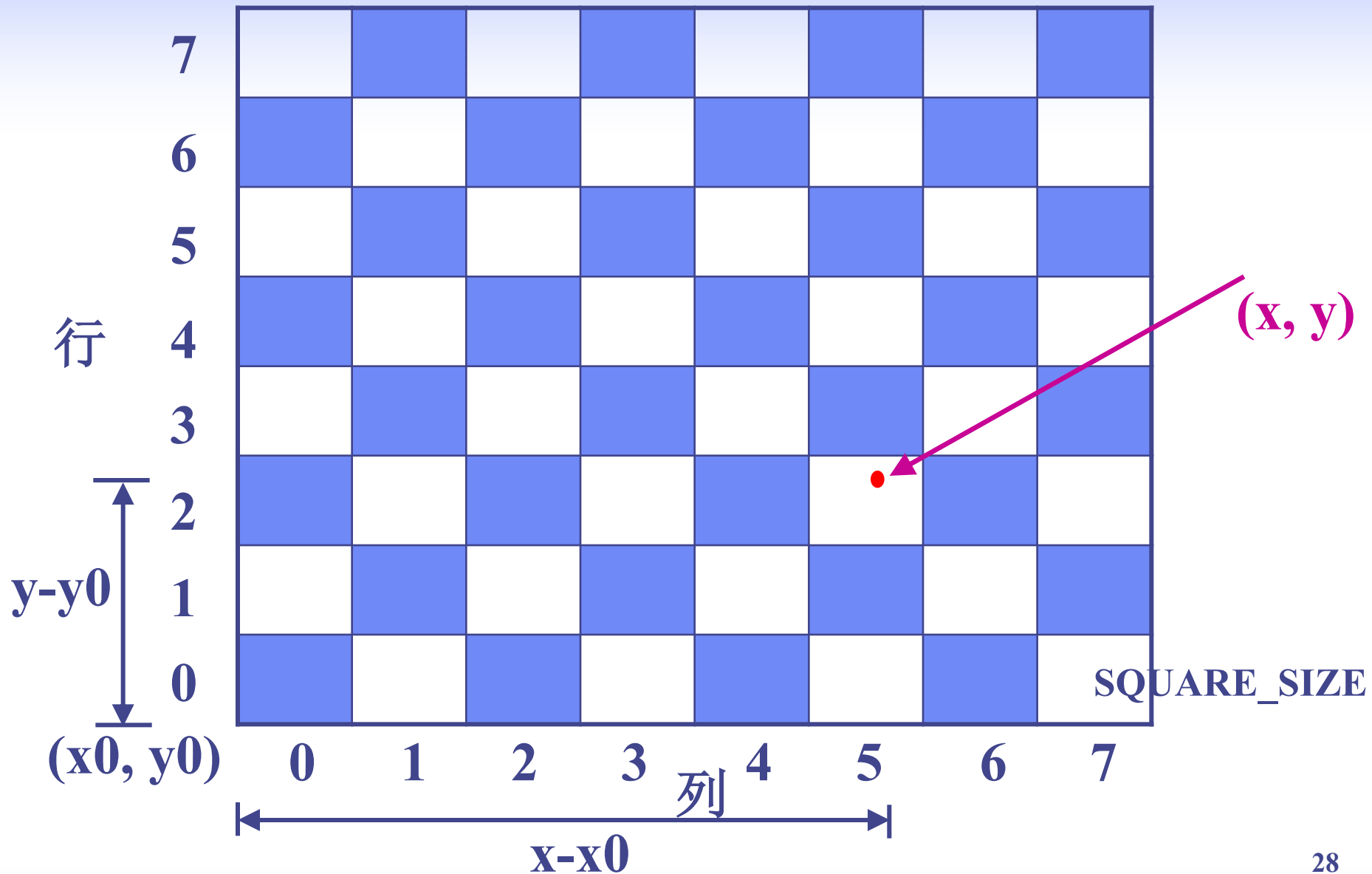
- ☺ 结构体：传值！

# 问题： 屏幕坐标到棋盘坐标的转换





# 问题分析:



```
struct ScreenCoor //屏幕坐标
{
    int x, y;
};
```

```
struct BoardCoor //棋盘坐标
{
    int row, col;
};
```

```

#define x0 40
#define y0 20
#define SQUARE_SIZE 10
void main()
{
    struct ScreenCoor s;
    struct BoardCoor b;
    cin >> s.x >> s.y;    // 输入屏幕坐标
    screen_to_board(______); // 函数调用
    cout << b.row << b.col << endl;
}

void screen_to_board(??screen, ??board)
{
    _____;
    _____;
}

```

```

#define x0 40
#define y0 20
#define SQUARE_SIZE 10
void main()
{
    struct ScreenCoor s;
    struct BoardCoor b;
    cin >> s.x >> s.y; // 输入屏幕坐标
    screen_to_board(s, &b); // 函数调用
    cout << b.row << ' ' << b.col << endl;
}

void screen_to_board(struct ScreenCoor screen,
                    struct BoardCoor *board)
{
    board->row = (screen.y - y0) / SQUARE_SIZE;
    board->col = (screen.x - x0) / SQUARE_SIZE;
}

```

# 本讲小结

- ◆ 结构体
- ◆ 结构体数组
- ◆ 结构体的传递



# 代码风格

◆ 空白

◆ 行宽

◆ 注释

◆ 命名

◆ 可参考 [Google C++ Style Guide](#)

# 空白

## ◆ 缩进

- 使用空格或者Tab，要一致
- 建议2个空格

## ◆ 符号

- 一般性规则
- 分支和循环
- 运算符

# 空白的一般性规则

```
int i = 0;  // Two spaces before end-of-line comments.
```

```
void f(bool b) {  // Open braces should always have a space before them.
```

```
...
```

```
int i = 0;  // Semicolons usually have no space before them.
```

```
// Spaces inside braces for braced-init-list are optional.  If you use them,  
// put them on both sides!
```

```
int x[] = { 0 };
```

```
int x[] = {0};
```

# 分支和循环中的空白

```
if (b) {           // Space after the keyword in conditions and loops.
} else {           // Spaces around else.
}
while (test) {}    // There is usually no space inside parentheses.
switch (i) {
for (int i = 0; i < 5; ++i) {
// Loops and conditions may have spaces inside parentheses, but this
// is rare. Be consistent.
switch ( i ) {
if ( test ) {
for ( int i = 0; i < 5; ++i ) {
// For loops always have a space after the semicolon. They may have a space
// before the semicolon, but this is rare.
for ( ; i < 5 ; ++i) {
...

```

# 运算符和空白

```
// Assignment operators always have spaces around them.
```

```
x = 0;
```

```
// Other binary operators usually have spaces around them, but it's
```

```
// OK to remove spaces around factors.  Parentheses should have no
```

```
// internal padding.
```

```
v = w * x + y / z;
```

```
v = w*x + y/z;
```

```
v = w * (x + z);
```

```
// No spaces separating unary operators and their arguments.
```

```
x = -5;
```

```
++x;
```

```
if (x && !y)
```

```
...
```

# 行宽

- ◆ 从便于阅读的角度出发
- ◆ 一般建议一行不要超过80个字符
- ◆ 一定不要超过100个字符
- ◆ 注意当一行太长时，使用合理的换行的位置

# 注释

- ◆ 简明，有意义
- ◆ 可以使用 `//` 和 `/* ... */`
- ◆ 一般要写注释的位置
  - 文件（在文件头部）
  - 函数（在函数的声明和定义之前）
  - 变量（在变量定义之前或者同一行）
  - 语句块（在语句块之前）

# 命名

## ◆ 使用英文命名

- 变量、类型用名词或名词性短语
- 函数用动词或动宾短语

## ◆ 常见的命名风格

- CamelCase：类型、函数
- snake\_case：变量
- BIG\_SNAKE\_CASE：宏



# 代码风格的目的和整体要求

- ◆ 增强代码可读性
- ◆ 要求前后一致