

信息学算法入门

内存地址

```

1  #include<iostream>
2  using namespace std;
3  int main(){
4      int x,y,z;
5      cout<<&x<<endl;
6      cout<<&y<<endl;
7      cout<<&z<<endl;
8      return 0;
9  }

```

0x提示后面
是十六进制

f是 15	e是 14	c是 12
----------	----------	----------

0x22fe4c

0x22fe48

0x22fe44

输出内存地址

十六进制表示

取地址运算符 &

& 翻译为：取地址

&x 翻译为：x的地址

数组地址1

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     int f[3];
5     cout<<&f[0]<<endl;
6     cout<<&f[1]<<endl;
7     cout<<&f[2]<<endl;
8     return 0;
9 }
```

请同学运行程序
结果和老师一样吗?

0x22fe40

0x22fe44

0x22fe48

.....	f[0]	f[1]	f[2]
0x22fe3c	0x22fe40	0x22fe44	0x22fe48	0x22fe4c

数组地址2

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int f[3];
5      cout<<&f[0]<<endl;      0x22fe40
6      cout<<&f[1]<<endl;      0x22fe44
7      cout<<&f[2]<<endl;      0x22fe48
8      cout<<f<<endl;          0x22fe40
9      cout<<f+1<<endl;        0x22fe44
10     cout<<f+2<<endl;        0x22fe48
11     return 0;
12 }
```

kkcoding.net

数组和地址

```
int f[3];
```

f 数组名**f**记录数组**0**号元素的**内存地址**

f+1 **f+1**记录数组**1**号元素的**内存地址**

f+2 **f+2**记录数组**2**号元素的**内存地址**

指针

引用运算符*

***** 翻译为：指向的内容

***p** 翻译为：p指向的变量值

int *p; 翻译为：定义p为指向整数的指针

p = &x; 翻译为：将x的地址赋值给p
也就是：p指向了x

p代表pointer指针

请预测输出结果

观察老师的输出结果

```
int x=8;
```

```
int *p;
```

```
p = &x;
```

```
cout<< x <<endl;
```

```
cout<< *p <<endl;
```

```
cout<< &x <<endl;
```

```
cout<< p <<endl;
```

p不是整数，p指向的是整数

定义p是指向整数的指针

p赋值为x的地址 p指向x

输出p指向的变量值

输出x的地址

输出p的值 就是x的地址

快快编程
kkcoding.net

请在电脑上完成程序再翻译这几行

老师检查

```
int x=8;
```

```
int *p;
```

```
p = &x;
```

```
cout<< x <<endl;
```

```
cout<< *p <<endl;
```

```
cout<< &x <<endl;
```

```
cout<< p <<endl;
```

定义p是指向整数的指针

p赋值为x的地址

p指向x

输出p指向的变量值

输出x的地址

输出p的值

就是x的地址

快快编程
kkcoding.net

指针

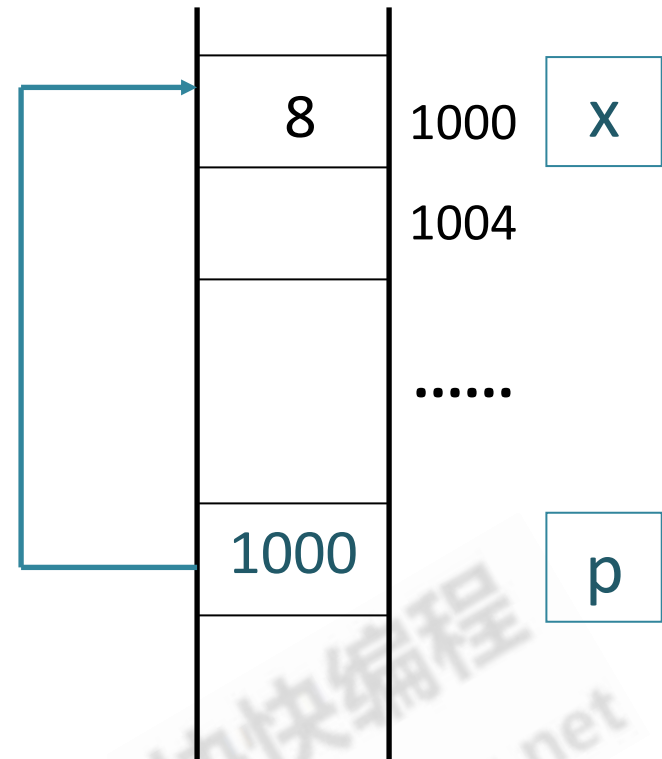
指针是**存储内存地址**的变量

```
int x=8;  
int *p;  
p = &x;
```

假设系统给x分配的地址是1000
那么p中保存的数据就是1000

两种访问方式:

- 直接访问变量x
- 间接访问指针p指向的变量内容



指针运算 &和*

```
int x=8;
```

```
int *p;
```

```
p = &x;
```

取地址运算符 & p赋值为x的地址

```
*p= 9;
```

引用运算符*

p指向的变量赋值为9

```
cout<<x<<endl;
```

等效于 x赋值为9

```

1  #include<iostream>
2  using namespace std;
3  int main() {
4      int x,y;
5      int *p;
6      x=3; y=4;
7      p=&x;
8      cout<<&x<<endl;
9      cout<<p<<endl;
10     cout<<*p<<endl;
11     *p=y+4;
12     cout<<x<<endl;
13     return 0;
14 }

```

p是指向整数的指针变量

p赋值为x的地址	p指向x
输出x的地址	
输出p	也就是输出x的地址
输出p指向的变量，也就是x	
p指向的变量x 赋值为y+4	

思考：为什么x变量的值最后变为8

指针演示

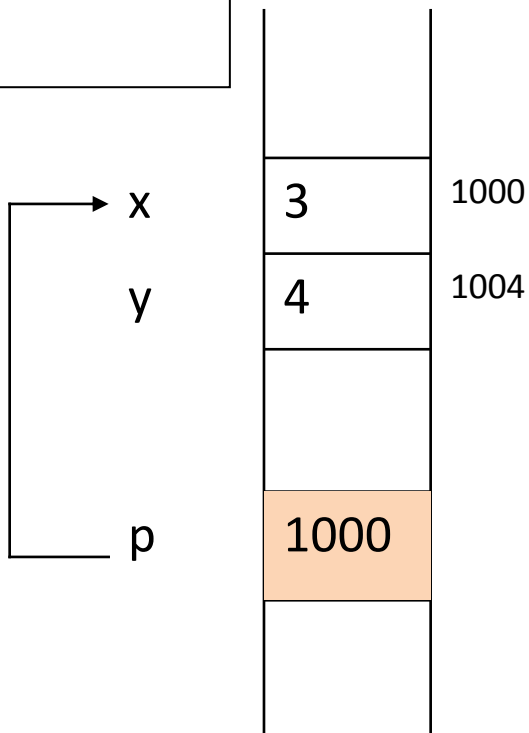
样例代码

```
int x,y,*p;
```

```
x=3;
```

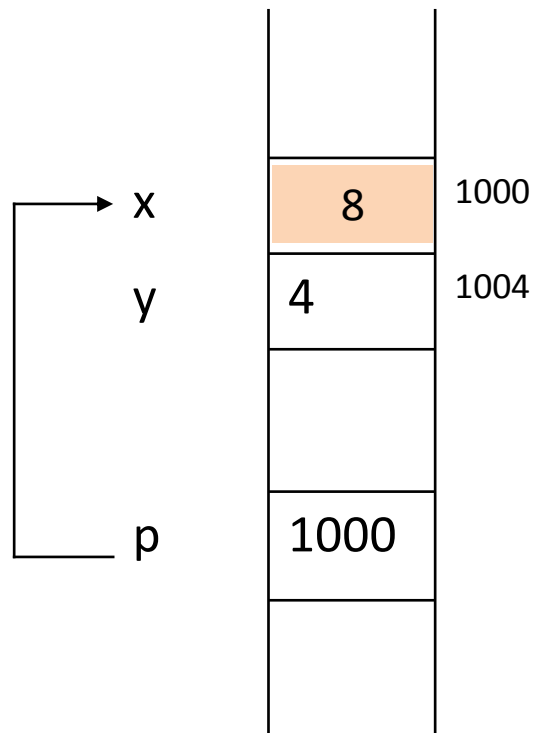
```
y=4;
```

```
p=&x;
```



执行代码

```
*p=y+4;
```



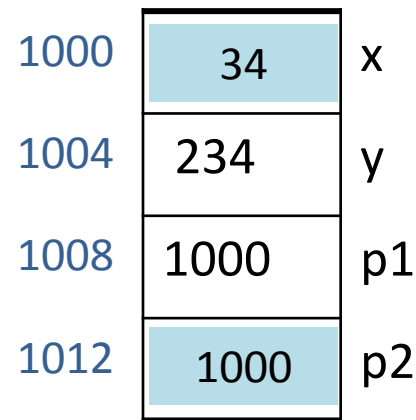
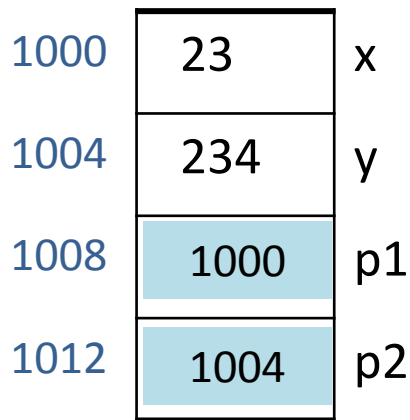
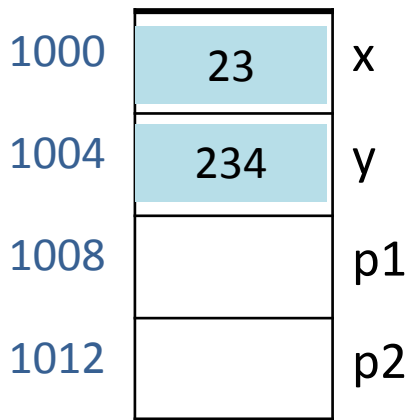
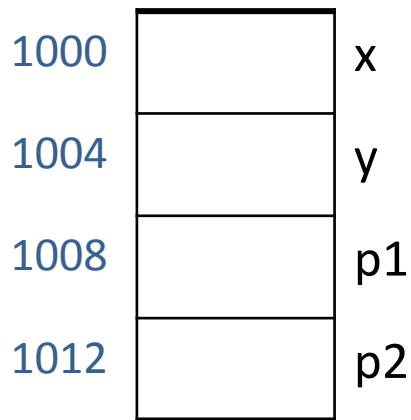
请理解内存中信息的变化

```
int x, y;  
int *p1,*p2;
```

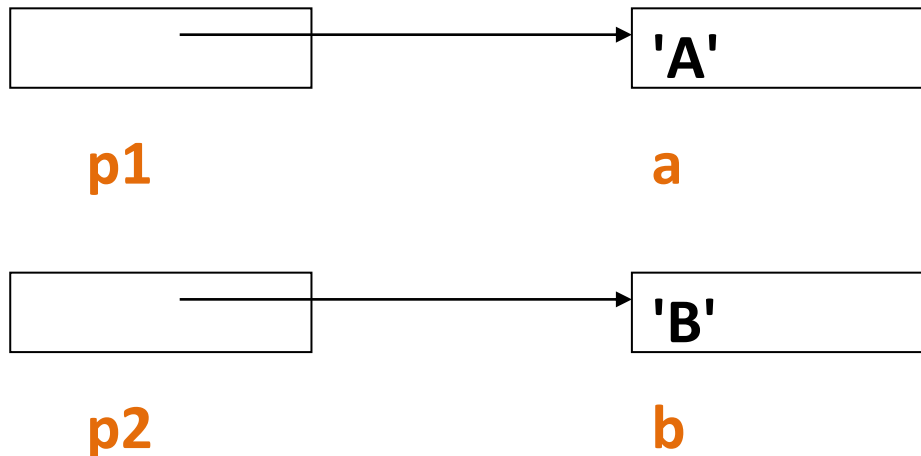
执行
`x=23;`
`y=234;`

执行
`p1=&x;`
`p2=&y;`

执行
`*p1=34;`
`p2=p1;`

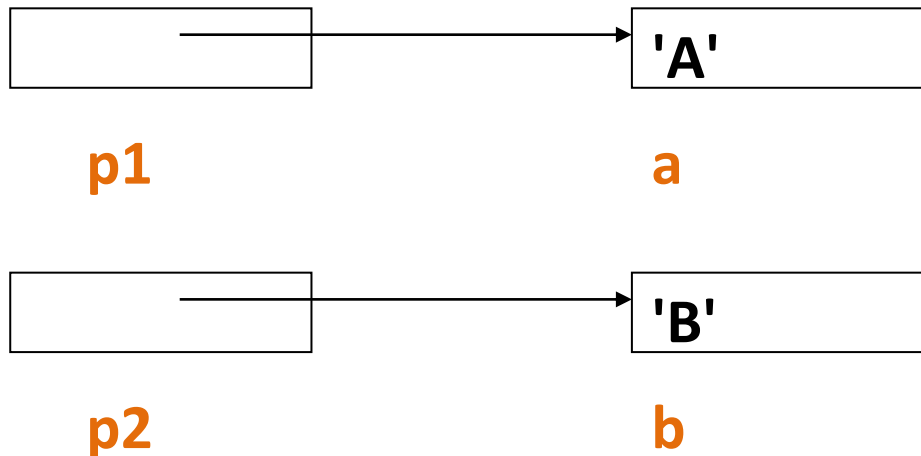



```
char *p1, *p2;  
char a='A';  
char b='B';  
p1=&a;  
p2=&b;
```



请描述p1=p2的结果

```
char *p1, *p2;  
char a='A';  
char b='B';  
p1=&a;  
p2=&b;
```



请描述 `*p1 = *p2` 的结果

指针和数组

请预测
输出结果

```
1 #include<iostream>
2 #include<string>
3 using namespace std;
4 string x[3]={"ha","wa","la"};
5 int main(){
6     cout<<x<<endl;
7     cout<<x+1<<endl;
8     cout<<x+2<<endl;
9     cout<<*x<<endl;
10    cout<<*(x+1)<<endl;
11    cout<<*(x+2)<<endl;
12    cout<<(x+1)[0]<<endl;
13    cout<<(x+1)[1]<<endl;
14    return 0;
15 }
```

存储
内存
地址

数组名x
也是指针

x+1
也是指针

x+2
也是指针

请口头翻译每一行

```
1  #include<iostream>
2  using namespace std;
3  int x[4]={5,6,7,8};
4  int main(){
5      int *p;
6      p=x+1;
7      cout<<p[0]<<endl;
8      cout<<p[1]<<endl;
9      cout<<p[2]<<endl;
10     return 0;
11 }
```

指针p也是数组名

将x数组1号元素的地址赋值给p

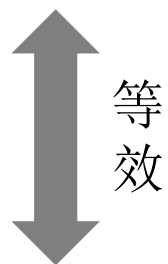
6

7

8

快快编程
kkcoding.net

指针变量名



数组名

都记录内存地址

顺序表

顺序表：数组实现

2	3	5	7
---	---	---	---	-------

存储要点

一段连续内存地址

顺序表算地址

定义数组 `int a[100];`
已知 `a[0]` 地址为 5000,
计算下列变量的地址

每个 `int` 变量
占 4 字节

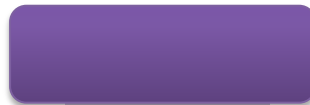
`a[0]`

5000

`a[1]`

5004

`a[8]`



`a[50]`



`a[99]`



给定随机编号

顺序表可以

$O(1)$ 时间定位

$O(1)$ 存取元素

基本数据类型

类型	占字节Byte	占比特bit	取值范围
int	4	32	$-2^{31} \sim 2^{31}-1$
long long	8	64	$-2^{63} \sim 2^{63}-1$
double	8	64	15位有效数字
char	1	8	$-128 \sim 127$
bool	1	8	0或1

最小的储存单位
是字节不是比特

1个字节=8个比特

基本数据类型

1个字节=8个比特

1个char变量占几个字节?几个比特?

1个int变量占几个字节?几个比特?

1个long long变量占几个字节?几个比特?

基本数据类型

1个字节=8个比特

int变量的范围是 $-2^{31} \sim 2^{31}-1$ ，能表示 2^{32} 个数

long long变量的范围是什么？能表示几个数

数组最值

max_element()

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4  int f[5]={3,1,5,1,2};
5  int main(){
6      cout<<max_element(f, f+5)<<endl;
7      cout<<max_element(f, f+5)-f<<endl;
8      cout<<*max_element(f, f+5)<<endl;
9      return 0;
10 }
```

算法库

max
最大

element
元素

请观察
输出结果

max_element()

```
1 #include<iostream>
2 #include<algorithm>
3 using namespace std;
4 int f[5]={3,1,5,1,2};
5 int main(){
6     cout<<max_element(f, f+5)<<endl;
7     cout<<max_element(f, f+5)-f<<endl;
8     cout<<*max_element(f, f+5)<<endl;
9     return 0;
10 }
```

算法库

地址

编号

数值

max
最大

element
元素

请观察
输出结果

数组的内存地址

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
------	----------	----------	----------	----------	----------	----------

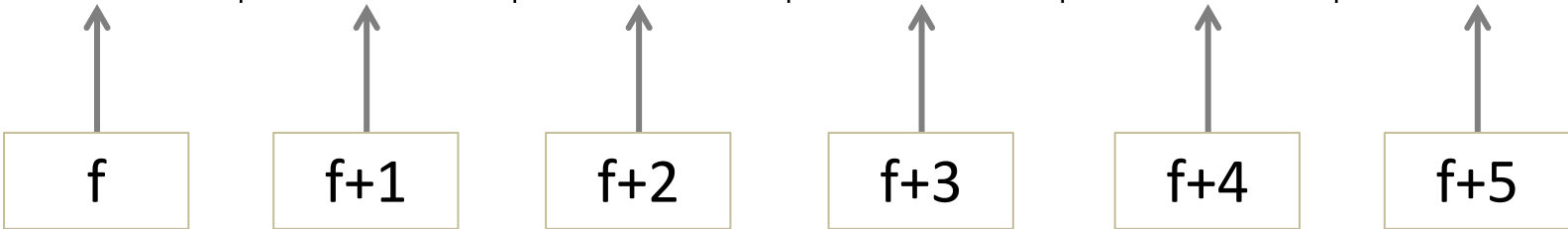


0x后面为16进制
每个地址放1字节

每个int变量
占4个字节

数组的内存地址

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	
内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
						
	f	f+1	f+2	f+3	f+4	f+5







快快编程
kkcoding.net

运行"数组地址"程序

数组的内存地址

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
						
	f	f+1	f+2	f+3	f+4	f+5

f为数组
开头地址

f+5正好
在数组外

数组最值的地址

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
	↑	↑	↑	↑	↑	↑
	f	f+1	f+2	f+3	f+4	f+5

`max_element(f, f+5)`
寻找从地址f开始, 在地址f+5之前
最大值的第一个地址
返回值为内存地址f+2

数组最值的编号

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
	↑	↑	↑	↑	↑	↑
	f	f+1	f+2	f+3	f+4	f+5

$\text{max_element}(f, f+5) - f$
是两个地址 $f+2$ 和 f 的距离
结果为整数 2, 代表数组编号

数组最值的数值

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

内存地址	0x472010	0x472014	0x472018	0x47201C	0x472020	0x472024
	↑	↑	↑	↑	↑	↑
	f	f+1	f+2	f+3	f+4	f+5

`*max_element(f, f+5)`
对应最大数值
星号*能取出地址内存放的数值

min_element()

```
1 #include<iostream>
2 #include<algorithm>
3 using namespace std;
4 int f[5]={3,1,5,1,2};
5 int main(){
6     cout<<min_element(f,f+5)<<endl;
7     cout<<min_element(f,f+5)-f<<endl;
8     cout<<*min_element(f,f+5)<<endl;
9     return 0;
10 }
```

min
最小

element
元素

请预测
输出结果

数组求最值

`min_element(f, f+n)`

翻译为：寻找从地址f开始, 在地址f+n之前
n个数里最小值的第一个地址

`min_element(f, f+n)`
`min_element(f, f+n) - f`
`*min_element(f, f+n)`

地址
编号
数值

时间复杂度 $O(n)$
实现方法还是逐个打擂台比大小

```
int f[5]={3,1,5,1,2};
```

数组编号	0	1	2	3	4	
数组元素值	3	1	5	1	2	

求f数组里的最小值

```
*min_element(f,f+5)
```

求f数组里前3个数里的最小值

```
*min_element(f,f+3)
```

求f数组里2号开始的最小值

```
*min_element(f+2,f+5)
```

求f数组里1号,2号,3号的最小值

```
*min_element(f+1,f+4)
```

f数组第1个最小值是几号

```
min_element(f,f+5)-f
```

f数组2号开始第1个最小值是几号

```
min_element(f+2,f+5)-f
```

快快编程作业

1797

878

1147

只要60分

拓展题

1447,676