



自学内容

C++对C的扩充

内容

□ 1.1 最简单的C++程序

□ 1.2 引用

□ 1.3 动态内存分配new和delete

1.1 最简单的C++程序

- ❑ 例1-1 编写程序，输入两个整数，输出其和。
- ❑ 用C语言来编写该程序。
- ❑ 用C++语言来实现该程序。
- ❑ 运行结果：

```
Input x, y:10 30
```

```
sum=40
```

```
Press any key to continue.
```

//C语言编写的程序

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    int x,y,sum;
```

```
    printf("Input x,y:");
```

```
    scanf("%d%d",&x,&y);
```

```
    sum=x+y;
```

```
    printf("sum=%d\n",sum);
```

```
}
```

//C++语言编写的程序

#include<iostream>

//变化1，头文件

using namespace std;

int main()

//变化2，main

{

int x,y,sum;

cout<<"Input x,y:";

//变化3，输出

cin>>x>>y;

//变化4，输入

sum=x+y;

cout<<"sum="<<sum<<endl;

return 0;

}

1.1 最简单的C++程序

- ❑ 例1-1 编写程序，输入两个整数，输出其和。
- ❑ 变化一 —— 输入输出头文件
- ❑ C++中采用cin和cout语句进行输入和输出
- ❑ cin和cout所用的头文件为： iostream
- ❑ C++标准库中的类和函数都在命名空间std中声明

```
#include<iostream>  
using namespace std;
```

1.1 最简单的C++程序

□例1-1 编写程序，输入两个整数，输出其和。

□变化二 —— main函数

```
int main ()  
{  
  
    .....  
    return 0;  
}
```

标准C++中main函数必须声明为int型。

如果程序正常执行，返回0；否则，返回-1。

1.1 最简单的C++程序

□ 变化三 —— 输出语句

□ 输出语句——cout:

cout << E1 <<E2<<...<<En;

□ << —— 流插入运算符，Ei——变量、常量或表达式；

□ 作用：从左向右将各表达式的值输出到显示器上当前光标位置。

□ 输出时，endl ——回车换行

1.1 最简单的C++程序

❑ 变化四—— 输入输出语句

❑ 输入语句——cin:

cin >> V1 >> V2>>...>>Vn;

❑ >> —— 流提取运算符，Vi——变量；

❑ 作用：从输入设备键盘读取数据依次赋值给V1、V2、...、Vn。

❑ 输入数据之间只能以空格、回车或制表符分隔！

❑ 输入数据的个数、类型与变量相一致。

//C++语言编写的程序

#include<iostream>
using namespace std;

//变化1，头文件

int main()

//变化2，main

{

int x,y,sum;

cout<<"Input x,y:"; //变化3，输出

cin>>x>>y; //变化4，输入

sum=x+y;

cout<<"sum="<<sum<<endl;

return 0;

}

1.1 最简单的C++程序

- 例1-1 编写程序，输入两个整数，输出其和。
- 自学练习1：这样的运行结果，如何编写？

```
Input x:10  
Input y:20  
(10+20)=30  
Press any key
```

1.1 最简单的C++程序

□说明：

□许多C++编译器为了与C兼容，保留了一些C的使用习惯。（部分编译器兼容）

□例如：

```
#include<iostream.h>  
void main ()  
{  
  
    .....  
  
}
```

1.2 引用

□ (一) 引用变量

□ 引用是为某个变量或对象起一个**别名**。格式：

数据类型 & 引用名 = 变量名；

□ 引用与其所代表的变量共享同一内存单元；

□ 引用在定义的同时必须初始化，初始化之后不能再引用成别的变量。



```
#include<iostream>
using namespace std;
int main()
{
```

```
    int x = 10;
```

```
    int & rx = x;
```

//rx是x的别名

```
    cout<<x<<" "<<rx<<endl;
```

```
    cout<<&x<<" "<<&rx<<endl;
```

```
    x=100;
```

```
    cout<<x<<" "<<rx<<endl;
```

```
    rx=200;
```

```
    cout<<x<<" "<<rx<<endl;
```

```
    return 0;
```

```
}
```

10 10

0012FF44 0012FF44

100 100

200 200

有什么
作用呢？

1.2 引用

□ (二) 引用作函数参数

□ 回顾：C语言中函数参数传递的两种方式：

1、按值传递



```
void swap ( int a,int b)      //按值传递
```

```
{
```

```
    int t;
```

```
    t=a; a=b; b=t;
```

```
}
```

```
int main()
```

```
{
```

```
    int i=3,j=5; cout<<i<<","<<j<<endl;
```

```
    swap(i,j);
```

```
    cout<<i<<","<<j<<endl;
```

```
    return 0;
```

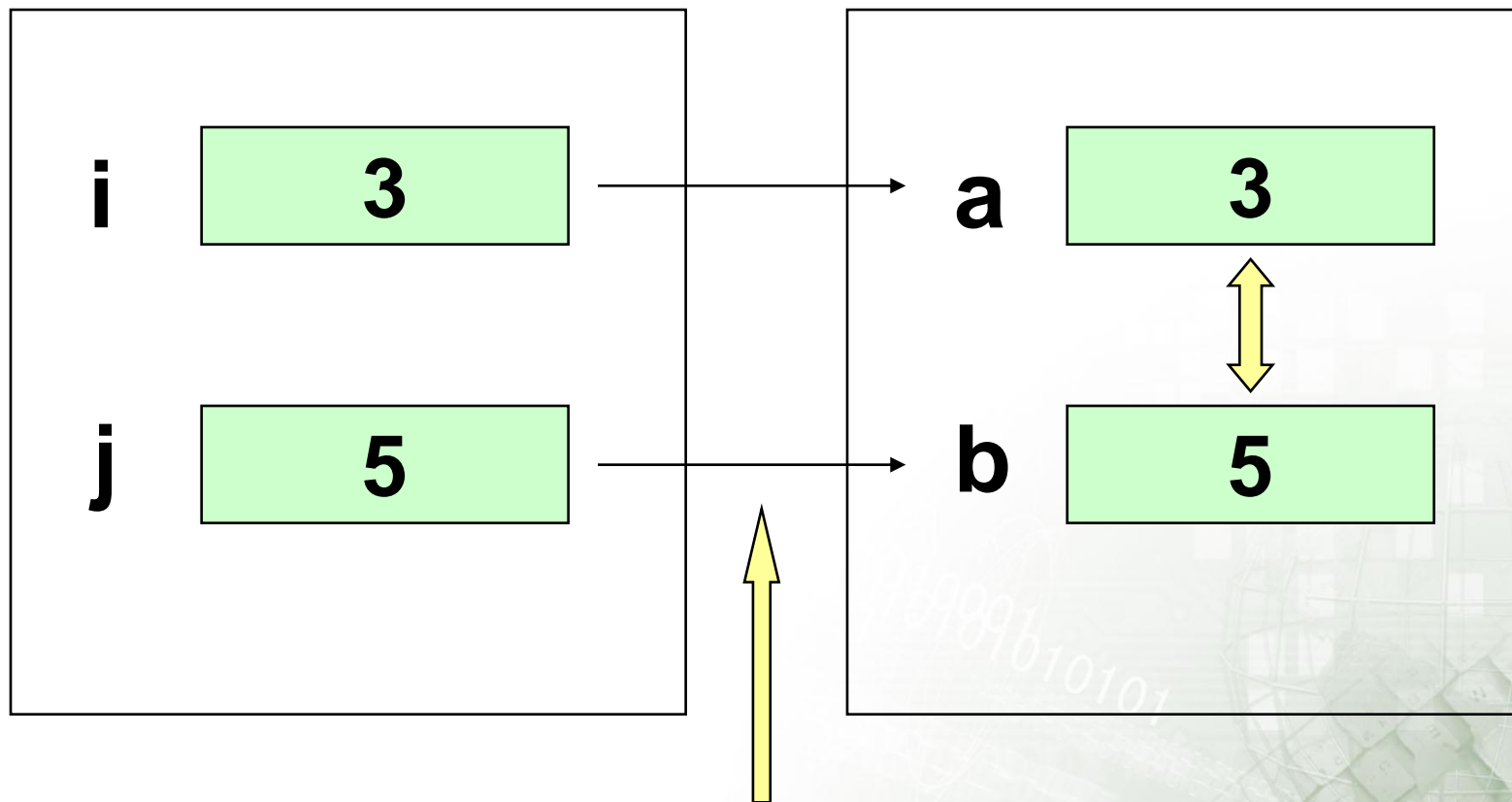
```
}
```

3, 5

3, 5

main函数

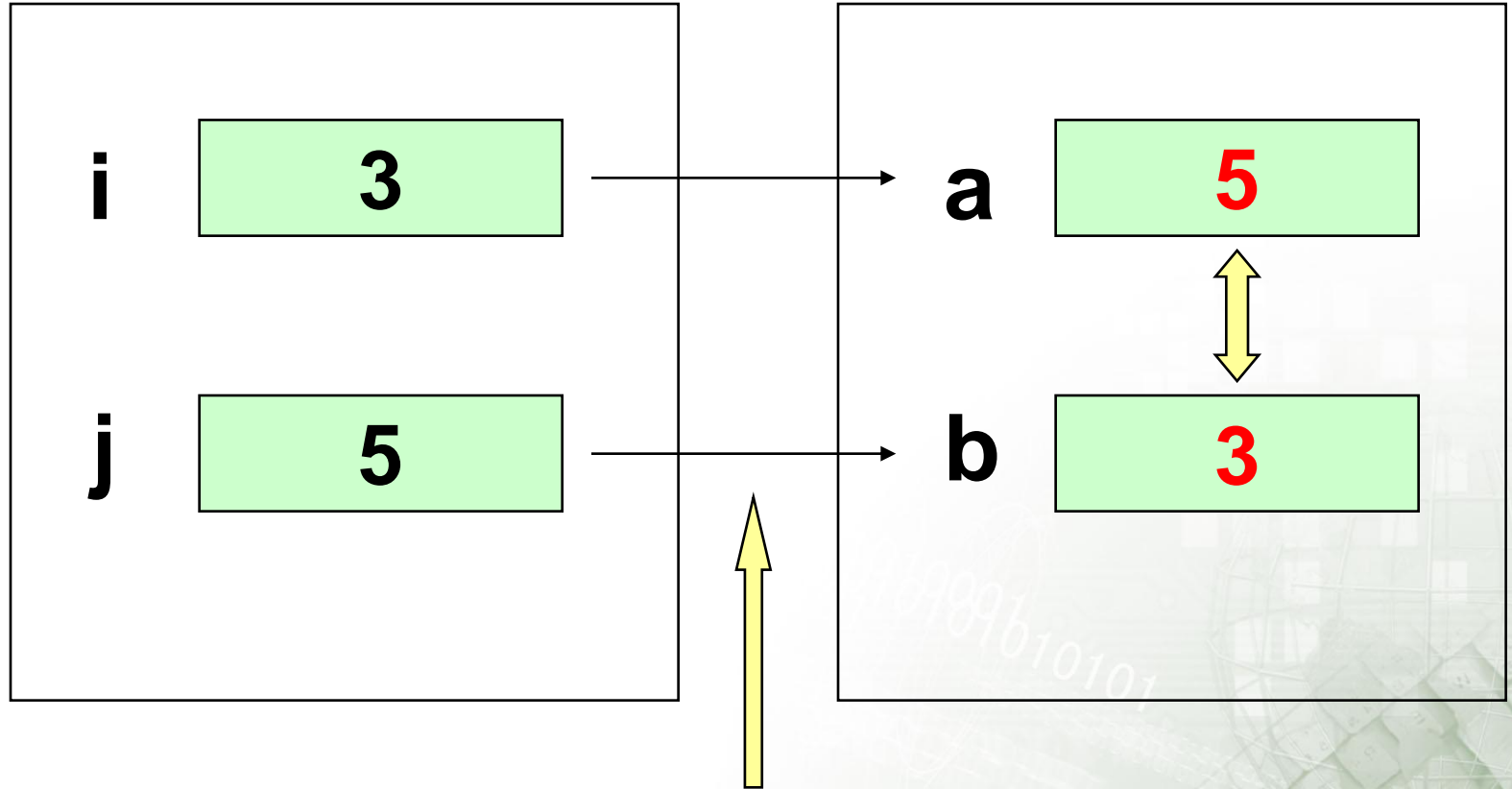
swap函数



传值

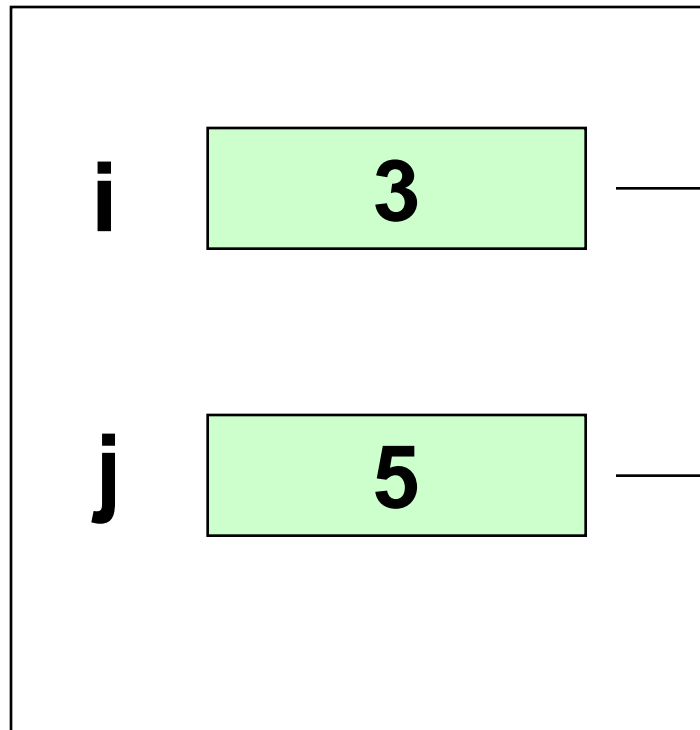
main函数

swap函数

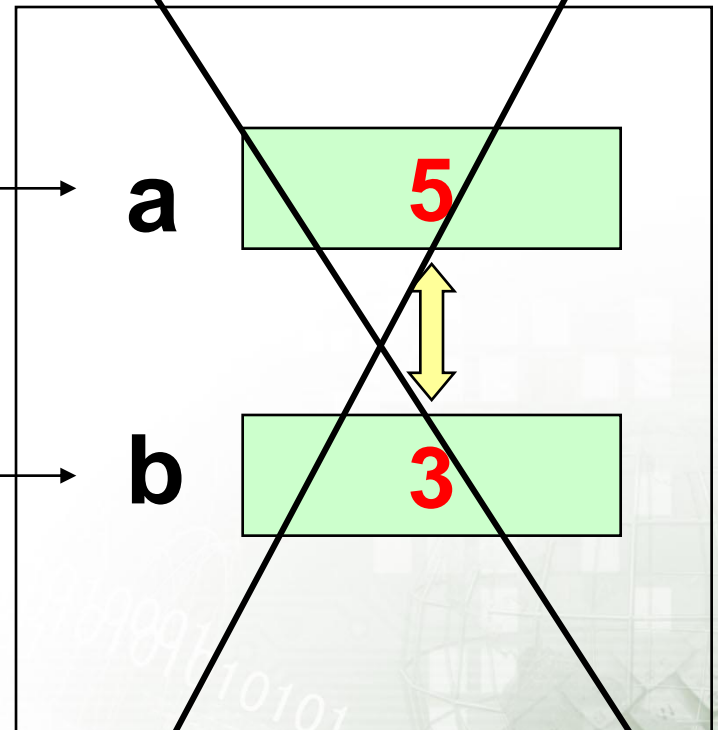


传值

main函数



swap函数



传值



```
void swap ( int a,int b)      //按值传递
```

```
{
```

```
    int t;
```

```
    t=a; a=b; b=t;
```

```
}
```

```
int main()
```

```
{
```

```
    int i=3,j=5; cout<<i<<","<<j<<endl;
```

```
    swap(i,j);
```

```
    cout<<i<<","<<j<<endl;
```

```
    return 0;
```

```
}
```

3, 5

3, 5

1.2 引用

□ (二) 引用作函数参数

□ 回顾：C语言中函数参数传递的两种方式：

1、按值传递

2、按地址传递



void swap (int *p1,int *p2) //按地址传递

```
{  
    int t;  
    t=*p1; *p1=*p2; *p2=t;  
}
```

```
int main()  
{
```

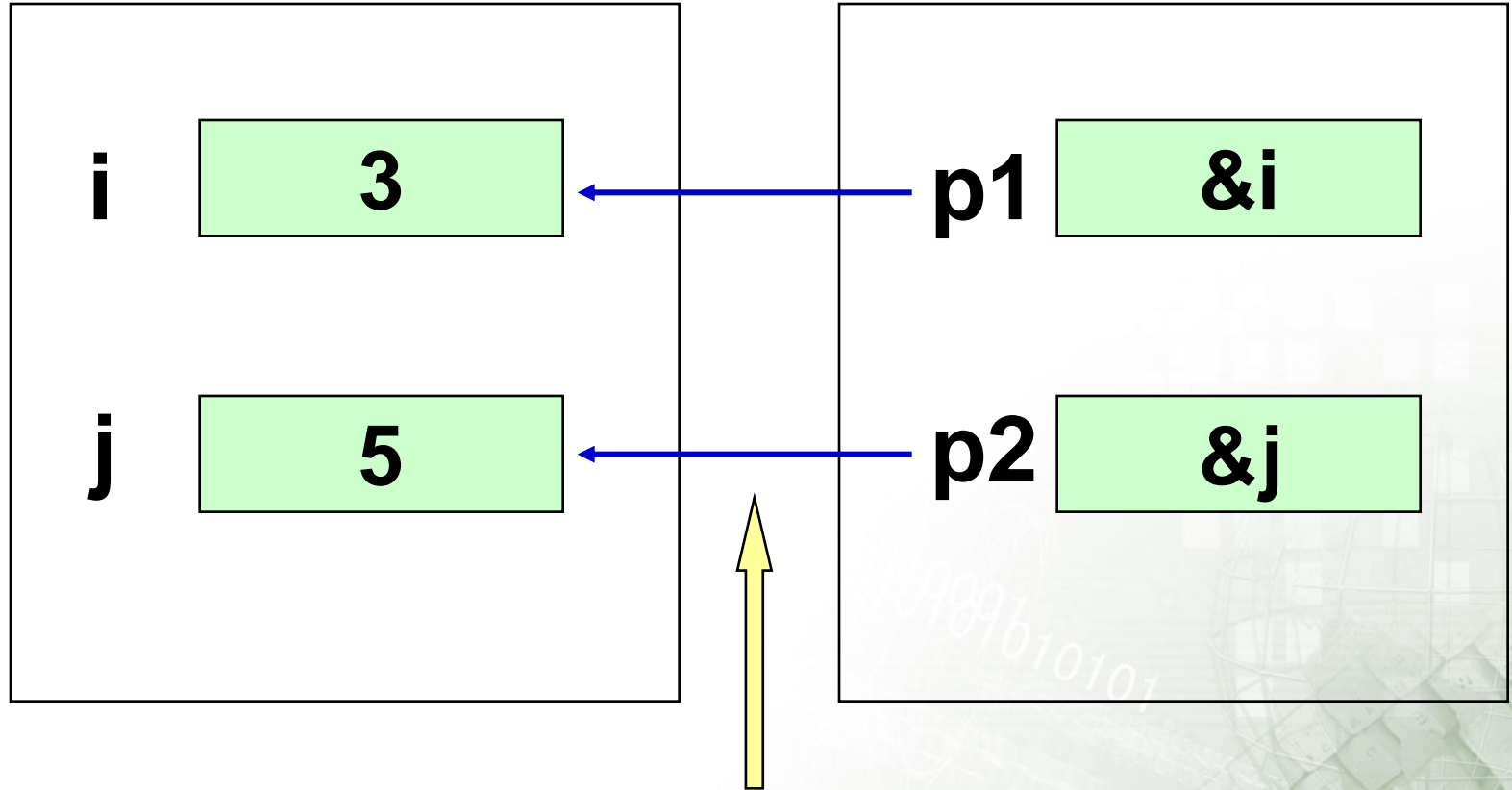
```
    int i=3,j=5; cout<<i<<","<<j<<endl;  
    swap(&i,&j);  
    cout<<i<<","<<j<<endl;  
    return 0;
```

```
}
```

3, 5
5, 3

main函数

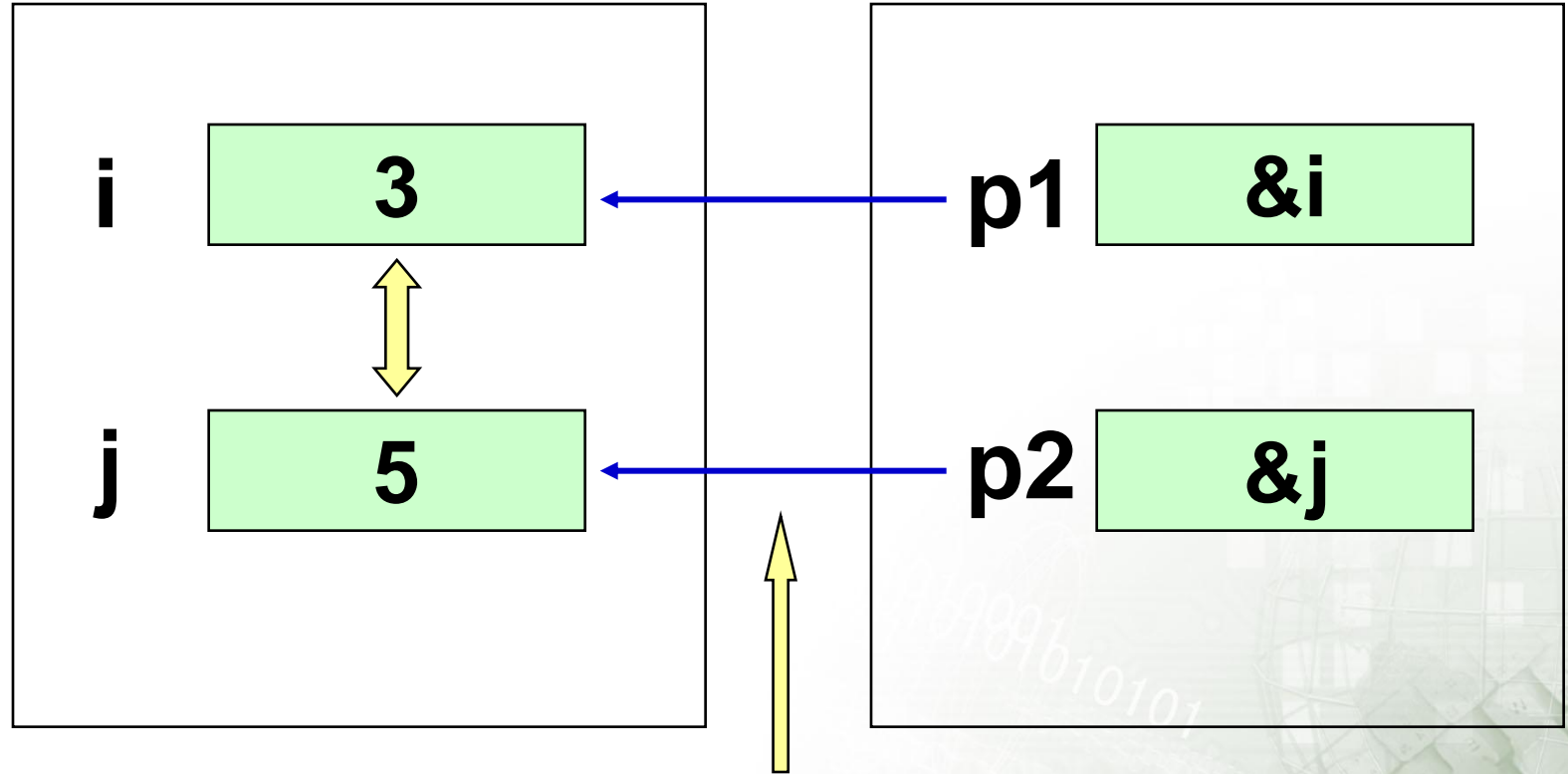
swap函数



传地址

main函数

swap函数

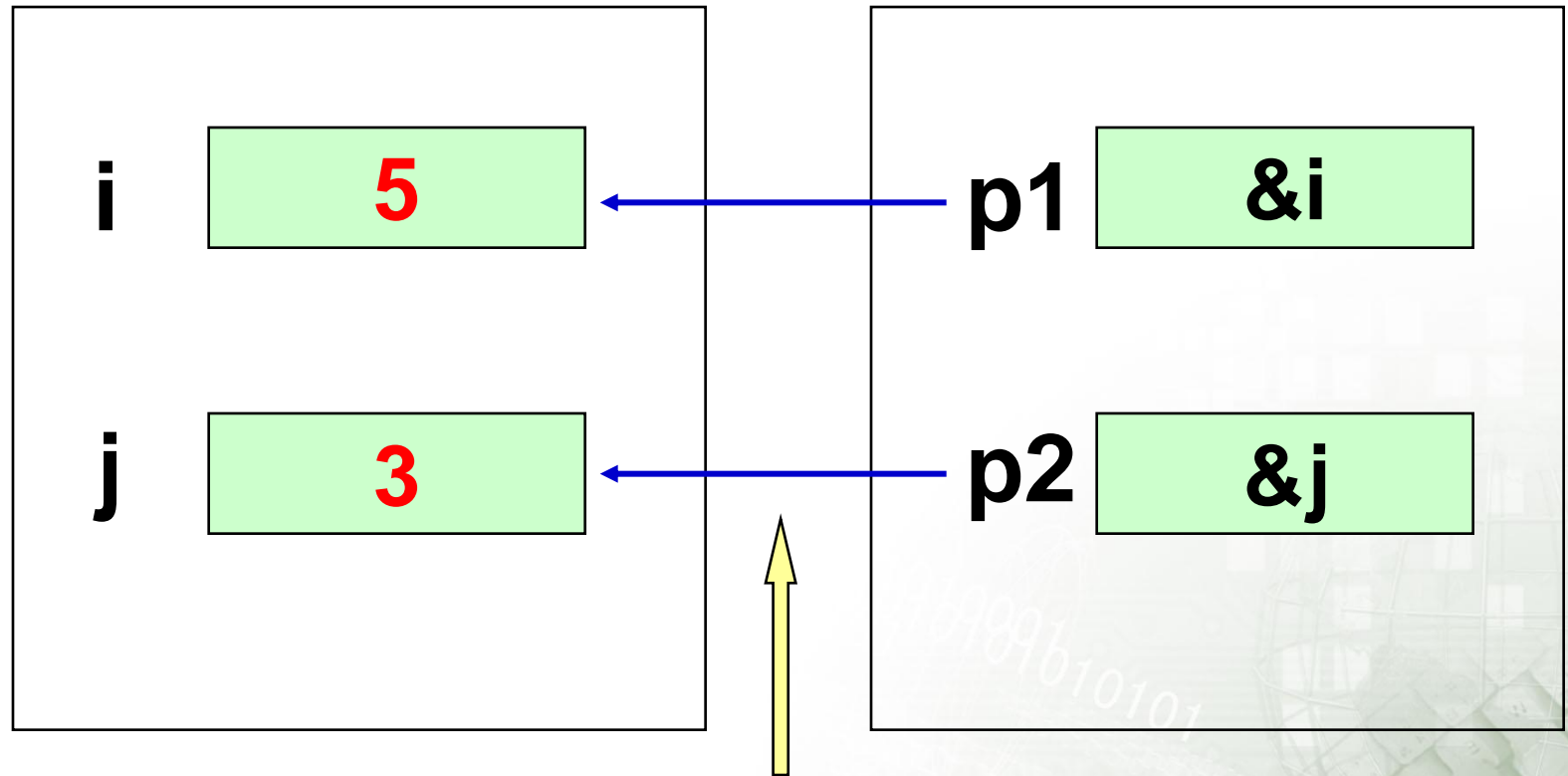


交换

$*p1 \Leftrightarrow *p2$

main函数

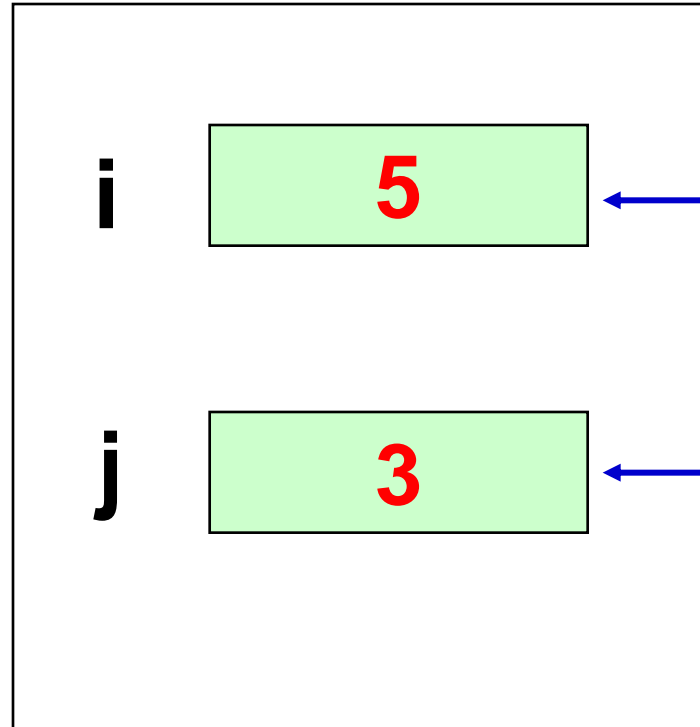
swap函数



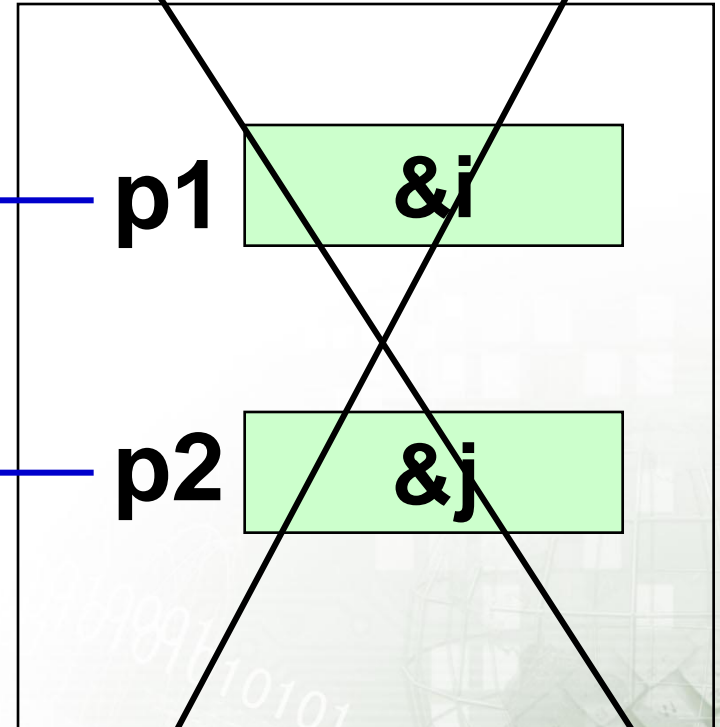
交换

***p1 <=> *p2**

main函数



swap函数



交换

$*p1 \Leftrightarrow *p2$



void swap (int *p1,int *p2) //按地址传递

```
{  
    int t;  
    t=*p1; *p1=*p2; *p2=t;  
}
```

```
int main()  
{
```

```
    int i=3,j=5; cout<<i<<","<<j<<endl;  
    swap(&i,&j);  
    cout<<i<<","<<j<<endl;  
    return 0;
```

```
}
```

3, 5
5, 3

1.2 引用

(二) 引用作函数参数

□ C++语言中函数参数传递的两种方式：

1、按值传递

2、按地址传递

3、按引用方式传递

1.2 引用

□ (二) 引用作函数参数

3、按引用方式传递

✓ 形参——引用变量

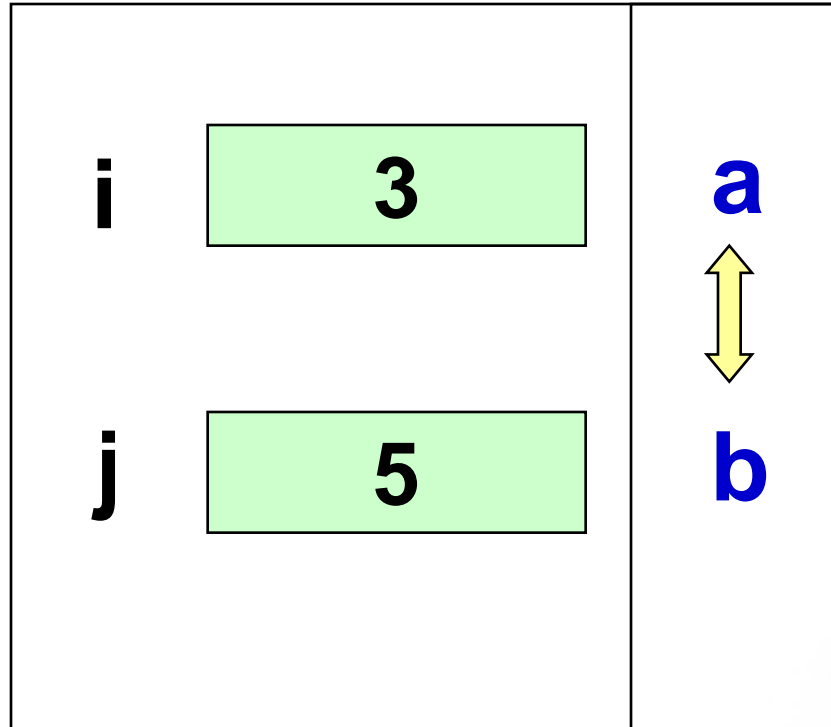
✓ 实参——变量

✓ 特点：形参变量是实参的引用，对形参值的改变直接影响实参。



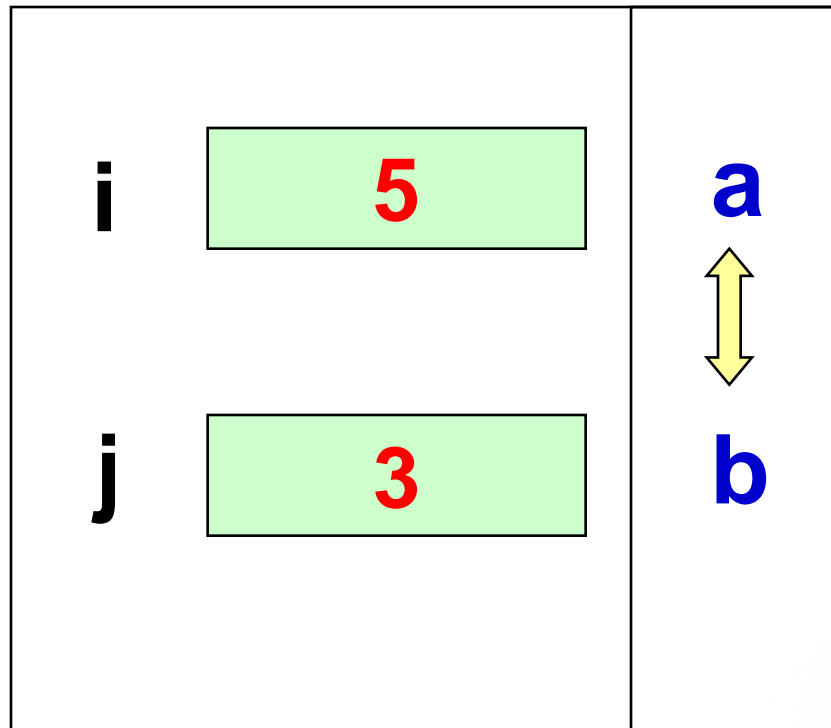
```
void swap ( int &a,int &b) //按引用传递
{
    int t;
    t=a; a=b; b=t;
}
int main()
{
    int i=3,j=5; cout<<i<<" "<<j<<endl;
    swap(i,j);
    cout<<i<<" "<<j<<endl;
    return 0;
}
```

main函数 swap函数



传引用

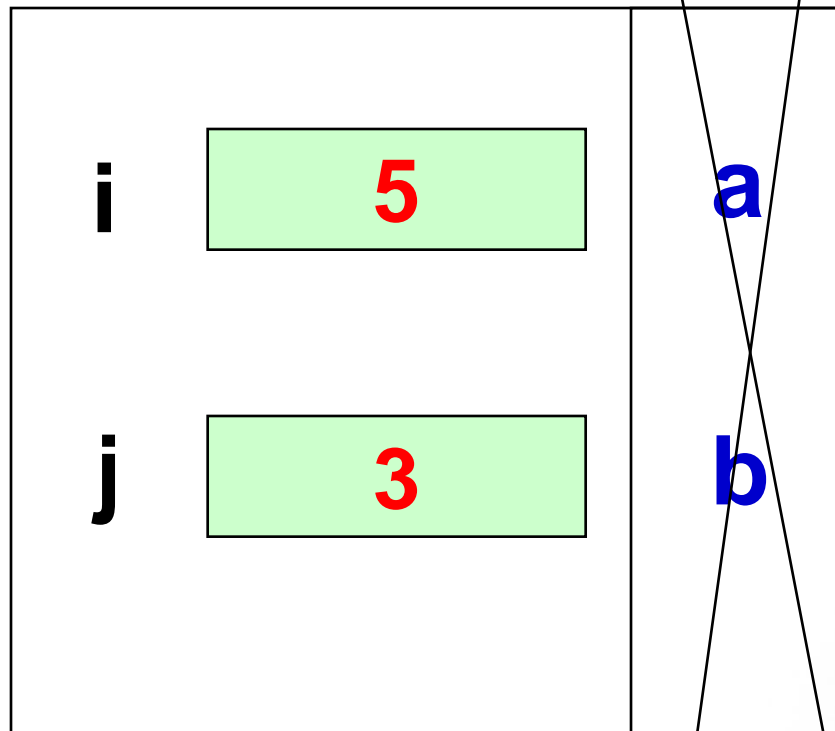
main函数 swap函数



传引用

main函数

swap函数



传引用



void swap (int &a,int &b) **//a,b是引用类型参数**

{ int t;

t=a; a=b; b=t;

**//交换a,b， 实际上就是
 交换 a,b所引用的参数**

}

int main()

{

int i=3,j=5; cout<<i<<","<<j<<endl;

**swap(i,j); //实参i,j传递给swap的形参
a,b， 相当于在swap中a,b是实参i， j的别名。**

cout<<i<<","<<j<<endl;

return 0;

}

3, 5

5, 3

1.2 引用

(二) 引用作函数参数

□ C++语言中函数参数传递的两种方式：

1、按值传递

2、按地址传递

3、按引用方式传递

□ 自学练习2：以下C语言程序如何改写为使用引用做参数的C++程序??

```
#include<stdio.h>
void Input(int *px)
{
    scanf("%d",px);
}
void main()
{
    int x;
    printf("Input x:");
    Input(&x);
    printf("x=%d\n",x);
}
```

1.3 动态内存分配new和delete

□（一）动态内存分配

□主要用于：内存需求不确定的场合。

□特点：根据需要，在程序运行过程中进行分配。

□例如：

链表的插入、删除等操作中，需要根据情况
随时分配结点，撤销结点。

1.3 动态内存分配new和delete

□ (一) 动态内存分配

□ C语言中支持动态内存分配的函数

1、 malloc函数

`void * malloc (unsigned int size)`

2、 calloc函数

`void * calloc(unsigned n, unsigned size)`

3、 free函数

`void free (void * p)`

1.3 动态内存分配new和delete

田 (二) 运算符new和delete

□ C++语言中提供运算符new和delete

✓ new —— 开辟空间

✓ delete —— 撤销所开辟的空间

□ 注意：

new所分配的空间一定要delete，否则会出现“内存泄露”。

两种使用方法：

1、动态分配和撤销**变量**

指针变量名 = new 类型 (初始化值) ;

delete 指针变量名 ;

2、动态分配和撤销**数组空间**

指针变量名 = new 类型 [长度] ;

delete [] 指针变量名 ;

□其中：该指针变量存储数组的首地址。

1.3 动态内存分配new和delete

(二) 运算符new和delete

□例1-3 存储1个整数100，然后输出。

□方法1：变量

□方法2：动态内存分配（用户自行管理）

//变量x，系统自动管理

int x;

x=100;

printf("整数： %d\n",x);

int *p;

//定义指针p

p=new int;

//动态分配1个int空间，把首地址给指针p

***p=100;**

//赋值

cout<<"整数： "<<(*p)<<endl;




//输出

delete p;

//使用完成后，动态撤销空间

1.3 动态内存分配new和delete

(二) 运算符new和delete

-  例1-4 定义数组存储1-10的整数，然后输出。
-  方法1：变量
-  方法2：动态内存分配（用户自行管理）

方法1：定义数组，系统自动管理

```
#include<iostream>
using namespace std;
int main()
{  int i,a[10];
    for(i=0;i<10;i++)
        a[i] = i+1;
    cout<<"The array are:"<<endl;
    for(i=0;i<10;i++)
        cout<< a[i] <<" ";
    cout<<endl;
    return 0;
}
```

The array are:

1 2 3 4 5 6 7 8 9 10

Press any key to continue

方法2：动态内存分配，用户自行管理

```
int main()
```

```
{
```

```
    int i, *a;
```

```
    a=new int[10];    //分配10个整数空间，首地址  
                      给指针a，a仍然相当于一个数组
```

```
    for(i=0;i<10;i++)    a[i] = i+1;
```

```
    cout<<"The array are:"<<endl;
```

```
    for(i=0;i<10;i++)    cout<< a[i] <<"  ";
```

```
    cout<<endl;
```

```
    delete [ ] a;    //使用完后，撤销数组空间
```

```
    return 0;
```

```
}
```

```
The array are:
```

```
1  2  3  4  5  6  7  8  9  10
```

```
Press any key to continue
```

两种使用方法：

1、动态分配和撤销**变量**

指针变量名 = new 类型 (初始化值) ;

delete 指针变量名 ;

**注意
这里**

2、动态分配和撤销**数组空间**

指针变量名 = new 类型 [**长度**] ;

delete [] 指针变量名 ;

□ **其中：长度还可以是变量。**

方法2：动态内存分配，用户自行管理

```
int main()
```

```
{ int i, n,*a;
```

```
    n=10;
```

```
    a=new int[n];    //分配n个整数空间，n可以是  
                    变量
```

```
    for(i=0;i<10;i++)        a[i] = i+1;
```

```
    cout<<"The array are:"<<endl;
```

```
    for(i=0;i<10;i++)        cout<< a[i] <<"  ";
```

```
    cout<<endl;
```

```
    delete [ ] a;    //使用完后，撤销数组空间
```

```
    return 0;
```

```
}
```

```
The array are:
```

```
1  2  3  4  5  6  7  8  9  10
```

```
Press any key to continue
```

□ 自学练习3：管理某个班学生的成绩，使用new和delete来操作。

程序框架：

- 1、输入学生的个数n；
- 2、定义一个存储n个学生成绩的数组
- 2、输入n个学生的成绩；
- 3、对n个学生的成绩输出。

总结

□ 扩充以下内容：

- ✓ 头文件，main，cin和cout语句；
- ✓ 引用变量，以及引用作函数参数；
- ✓ new和delete使用

□ 作业：

完成3个自学练习的程序。