§6.指针基础 - 画内存图并分析

要求：

1、模仿第06模块PDF课件中(P.15-18)的样式，画出下列每小题每一步执行的内存分配及指向图示，分析为什么得到最后的结果。

★ PDF课件的P.30（如何同时得到周长和面积）

★ PDF课件的P.31（为什么无法进行交换）

★ PDF课件的P.32（为什么会出现错误，导致错误的关键语句是哪一句）

2、每个语句要画一张内存状态图，每小题都是4张图

★ 第1张初始内存分配图附件已给出

3、不允许手写、手写后贴图

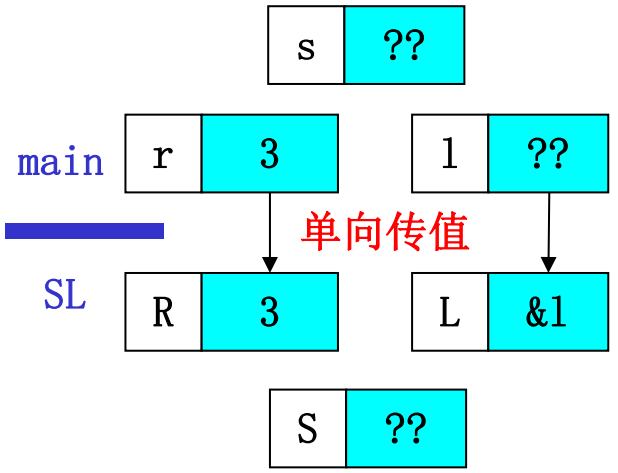
4、转换为pdf后在"实验报告"中提交（5.12前）

§6. 指针基础 – 画内存图并分析

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

#define PI 3.14159

double SL(double R, double *L)
{   double S;
    S = PI*R*R;
    *L = 2*PI*R;
    return S;
}
int main()
{   double s,l, r=3;
    s=SL(r,&l);
    cout << "s=" << s << endl;
    cout << "l=" << l << endl;
}
```

函数执行后同时得到周长及面积
周长：指针变量做形参方式
面积：函数返回值方式
注:函数的return只能带一个返回值!!

s=28.2743
l=18.8495



初始内存分配如图所示
请自行画出SL中三句话
执行时内存的变化
理解最后的输出结果

§6. 指针基础 – 画内存图并分析

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

#define PI 3.14159

double SL(double R, double *L)
{   double S;
    S = PI*R*R;
    *L = 2*PI*R;
    return S;
}
int main()
{   double s,l, r=3;
    s=SL(r,&l);
    cout << "s=" << s << endl;
    cout << "l=" << l << endl;
}
```

函数执行后同时得到周长及面积

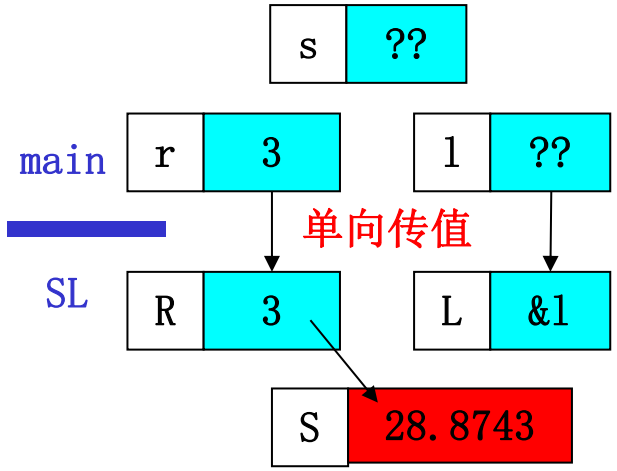
周长：指针变量做形参方式

面积：函数返回值方式

注:函数的return只能带一个返回值!!

s=28.2743

l=18.8495



§6. 指针基础 – 画内存图并分析

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

#define PI 3.14159

double SL(double R, double *L)
{   double S;
    S = PI*R*R;
    *L = 2*PI*R;
    return S;
}
int main()
{   double s,l, r=3;
    s=SL(r,&l);
    cout << "s=" << s << endl;
    cout << "l=" << l << endl;
}
```

函数执行后同时得到周长及面积

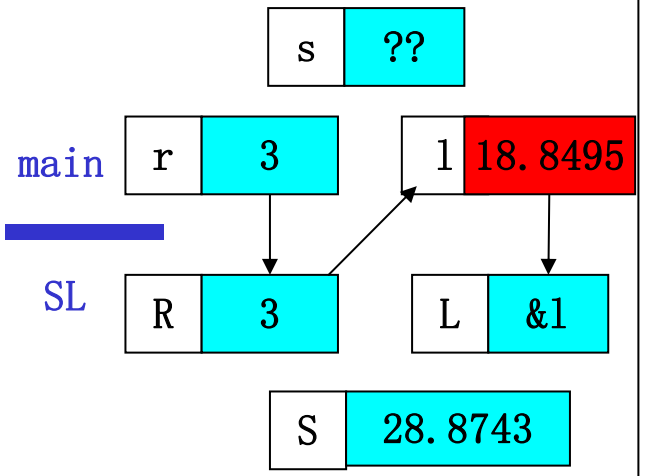
周长：指针变量做形参方式

面积：函数返回值方式

注:函数的return只能带一个返回值!!

s=28.2743

l=18.8495



§6. 指针基础 – 画内存图并分析

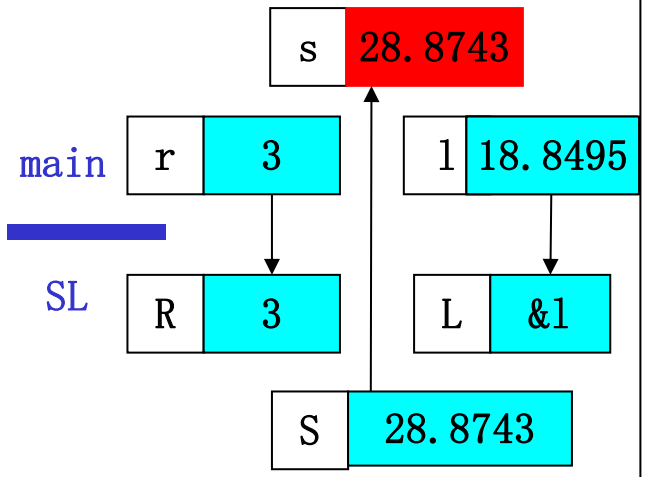
```
#include <iostream>
using namespace std;

#define PI 3.14159

double SL(double R, double *L)
{   double S;
    S = PI*R*R;
    *L = 2*PI*R;
    return S;
}
int main()
{   double s,l, r=3;
    s=SL(r,&l);
    cout << "s=" << s << endl;
    cout << "l=" << l << endl;
}
```

函数执行后同时得到周长及面积

周长：指针变量做形参方式

面积：函数返回值方式

注:函数的return只能带一个返回值!!



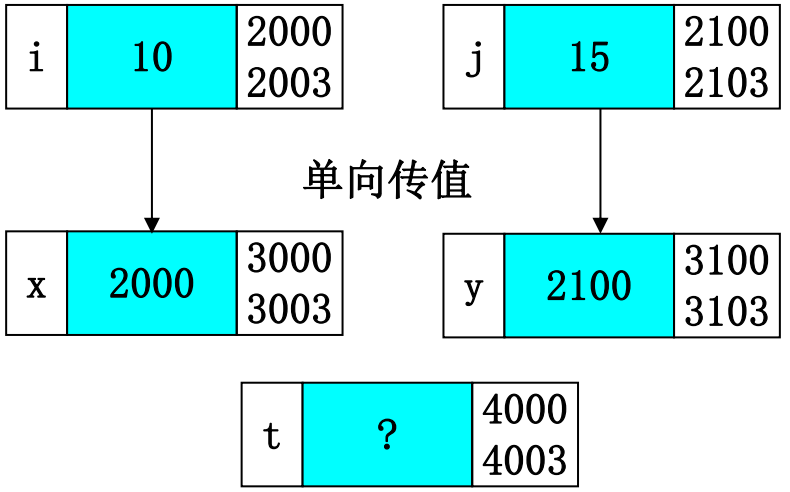
s=28.2743

l=18.8495

传回后两个实参都被改变，正确输出。

§6. 指针基础 – 画内存图并分析

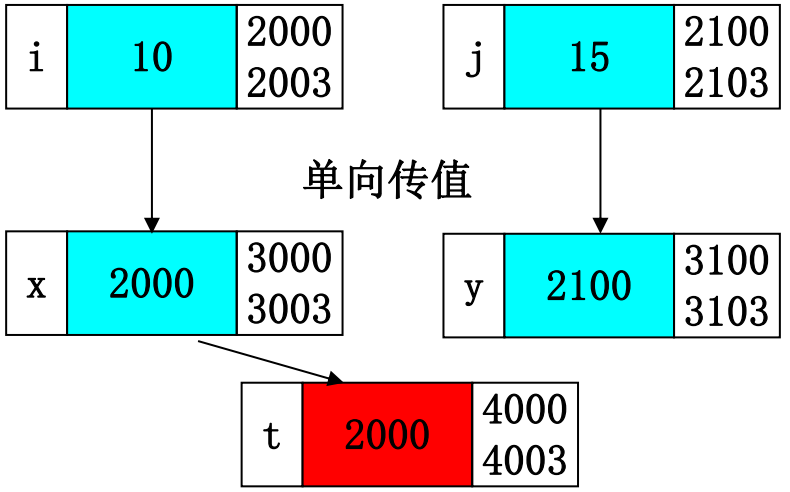
```
void swap(int *x, int *y)
{    int *t;
     t = x;
     x = y;
     y = t;
}

int main()
{
     int i=10, j=15;
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
     swap(&i,&j);
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
}
```



初始内存分配如图所示
请自行画出swap中三句话
执行时内存的变化
理解为什么无法交换

§6. 指针基础 – 画内存图并分析

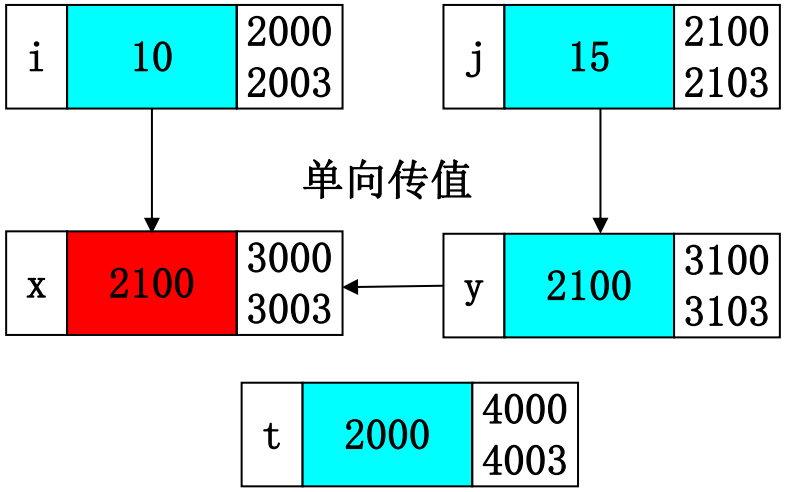
```
void swap(int *x, int *y)
{    int *t;
     t = x;
     x = y;
     y = t;
}

int main()
{
     int i=10, j=15;
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
     swap(&i,&j);
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
}
```



初始内存分配如图所示
请自行画出swap中三句话
执行时内存的变化
理解为什么无法交换

§6. 指针基础 – 画内存图并分析

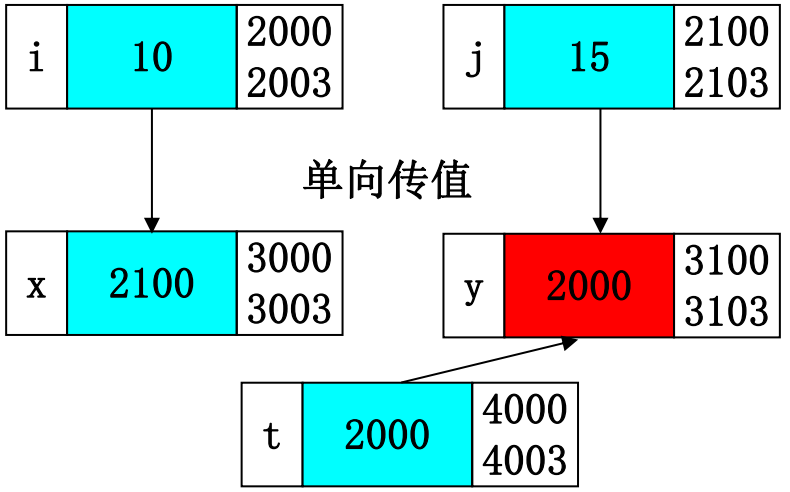
```
void swap(int *x, int *y)
{    int *t;
     t = x;
     x = y;
     y = t;
}

int main()
{
     int i=10, j=15;
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
     swap(&i,&j);
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
}
```



初始内存分配如图所示
请自行画出swap中三句话
执行时内存的变化
理解为什么无法交换

§6. 指针基础 – 画内存图并分析

```
void swap(int *x, int *y)
{    int *t;
     t = x;
     x = y;
     y = t;
}

int main()
{
     int i=10, j=15;
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
     swap(&i,&j);
     cout << "i=" << i << " j=" << j << endl;    i=10 j=15
}
```



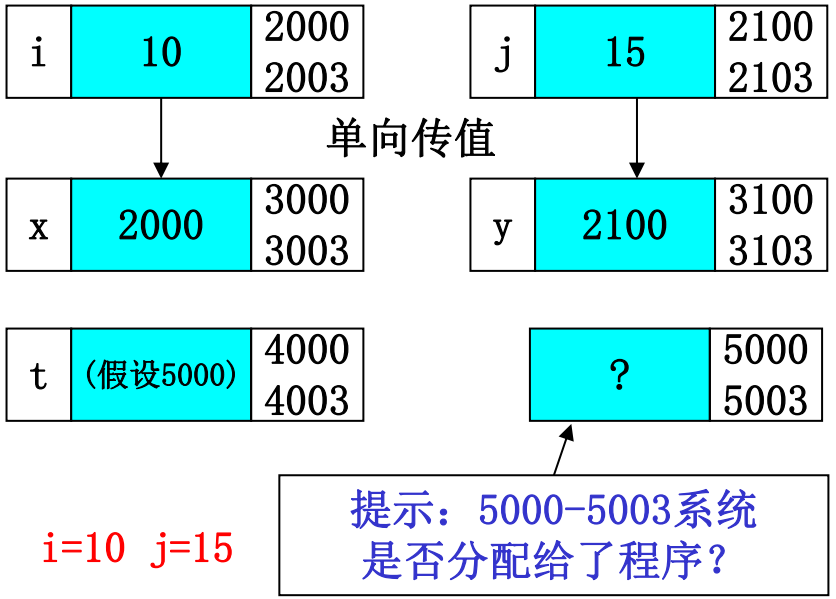
初始内存分配如图所示
请自行画出swap中三句话
执行时内存的变化
理解为什么无法交换

只改变了函数中的形参，没有改变真正的实参，
故交换失败。

§6. 指针基础 – 画内存图并分析

```
void swap(int *x, int *y)
{    int *t;
     *t = *x;
     *x = *y;
     *y = *t;
}

int main()
{
     int i=10, j=15;
     cout << "i=" << i << " j=" << j << endl;
     swap(&i,&j);
     cout << "i=" << i << " j=" << j << endl;
}
```

VS2019编译报错
-使用了未初始化的局部变量t
其它编译器可能可以运行
初始内存分配如图所示，请自行画出swap中三句话执行时内存的变化，理解为什么出现严重错误
另1：哪句是错误的关键？
另2：int *t 改为 int tt, *t;
t = &tt;
为什么就正确了？



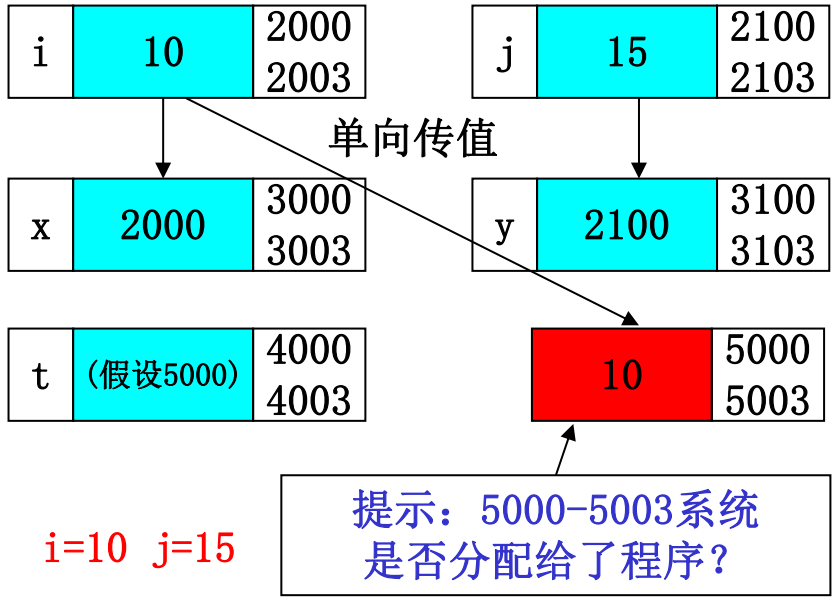
i=10 j=15

i=15 j=10

或 死机或其它非正常现象

§6. 指针基础 – 画内存图并分析

```
void swap(int *x, int *y)
{     int *t;
      *t = *x;
      *x = *y;
      *y = *t;
}

int main()
{
      int i=10, j=15;
      cout << "i=" << i << " j=" << j << endl;
      swap(&i,&j);
      cout << "i=" << i << " j=" << j << endl;
}
```



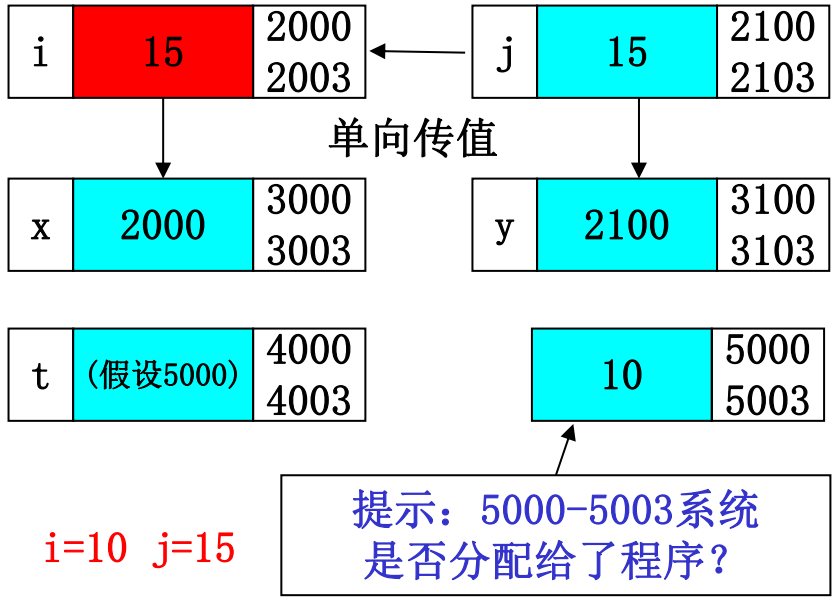
i=10 j=15

i=15 j=10 系统没有分配给程序

或 死机或其它非正常现象

§6. 指针基础 – 画内存图并分析

```
void swap(int *x, int *y)
{     int *t;
      *t = *x;
      *x = *y;
      *y = *t;
}

int main()
{
      int i=10, j=15;
      cout << "i=" << i << " j=" << j << endl;
      swap(&i,&j);
      cout << "i=" << i << " j=" << j << endl;
}
```



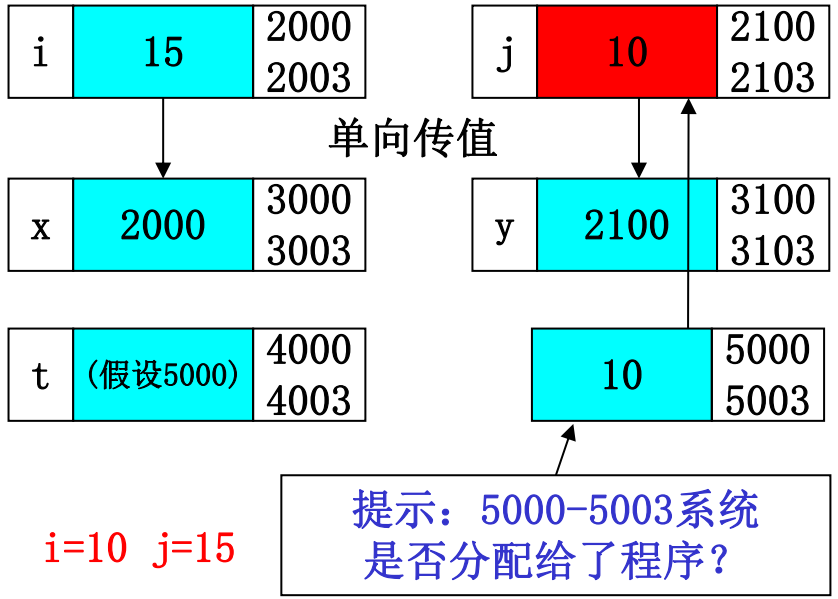
i=10 j=15

i=15 j=10 系统没有分配给程序

或 死机或其它非正常现象

§6. 指针基础 – 画内存图并分析

```
void swap(int *x, int *y)
{     int *t;
      *t = *x;
      *x = *y;
      *y = *t;
}

int main()
{
      int i=10, j=15;
      cout << "i=" << i << " j=" << j << endl;
      swap(&i,&j);
      cout << "i=" << i << " j=" << j << endl;
}
```



i=10 j=15

i=15 j=10 系统没有分配给程序

或 死机或其它非正常现象

VS2019编译报错
-使用了未初始化的局部变量t
其它编译器可能可以运行
初始内存分配如图所示，请自行画出swap中三句话执行时内存的变化，理解为什么出现严重错误
另1：哪句是错误的关键？
另2：int *t 改为 int tt, *t;
t = &tt;
为什么就正确了？

答：1. 第一句是错误的关键，因为指针t未初始化，程序没有分配初始内存，t指向的地址未知，编译器会报错。

2. int tt在函数内部定义了一个int型变量，系统分配了4字节内存，之后将tt的地址赋给指针变量t，此时t指向的地址已经分配给了程序，故不会报错。