北京大学

《计算概论A》课程 程序设计部分

指针（3）

李　戈

北京大学信息科学技术学院

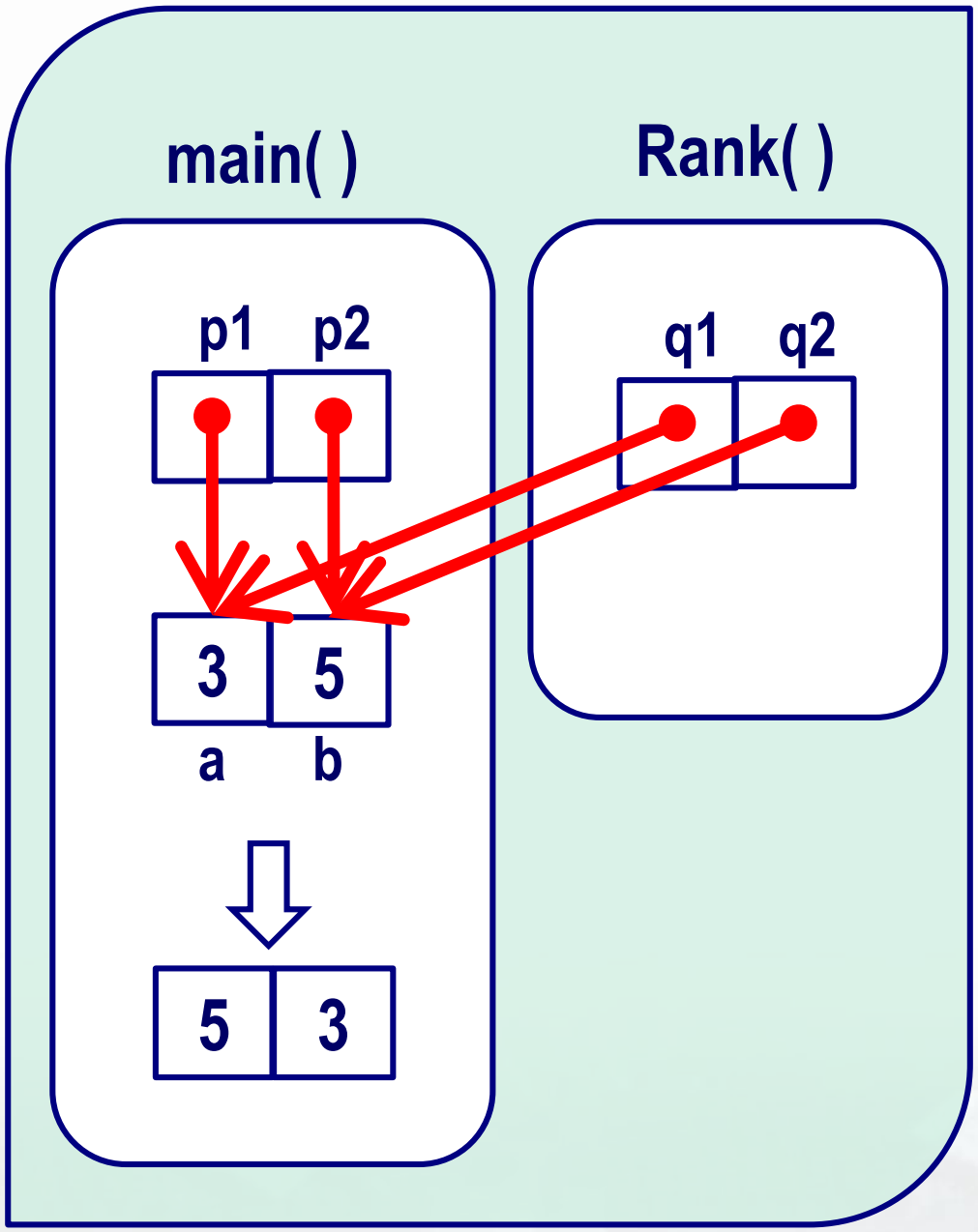
lige@sei.pku.edu.cn

本节内容

- 指针与函数
  - 指针用做函数参数
    - 如何“限制”指针的功能
  - 指针用做函数返回值
    - 静态局部变量

指针变量做函数参数

```
#include<iostream>
using namespace std;
void Rank(int *q1, int *q2)
{
    int temp;
    if (*q1 < *q2)
    {
        temp = *q1;
        *q1 = *q2;
        *q2 = temp;
    }
}
int main()
{
    int a, b, *p1, *p2;
    cin >> a >> b;
    p1 = &a; p2 = &b;
    Rank(p1, p2);
    cout << a << " " << b << endl;
    return 0;
}
```



数组名做函数参数

- 可否将数组名作为实参赋给指针型形参？可以！

```
#include<iostream>
using namespace std;
int main()
{
    int a[10] =
        {1,2,3,4,5,6,7,8,9,10};
    sum(a,10);
    return 0;
}
```

```
void sum(int *p, int n)
{
    int total = 0;
    for(int i=0;i<n;i++)
    {
        total += *p++;
    }
    cout<<total<<endl;
}
```

多维数组名做函数参数

例：有一个3×4的矩阵，求所有元素中的最大值。

```
int maxvalue(                    )
{
    int max = p[0][0];
    for(int i=0; i<3; i++)
        for(int j=0; j<4; j++)
            if(p[i][j]>max)
                max = p[i][j];
            return max;
}
int main( )
{
    int a[3][4] = {{1,3,5,7},{9,11,13,15},{2,4,6,8}};
    cout<<"The Max value is "<<maxvalue(a);
    return 0;
}
```

“数组名”做形参 可以吗?

```
#include<iostream>
using namespace std;
int sum(int array[], int n)
{
    for (int i = 0; i < 10 - 1; i++)
    {
        *(array + 1) = *array + *(array + 1);
        array++;
    }
    return *array;
}
int main()
{
    int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    cout << sum(a, 10);
    return 0;
}
```

可以！

C++编译器

将形参数组名

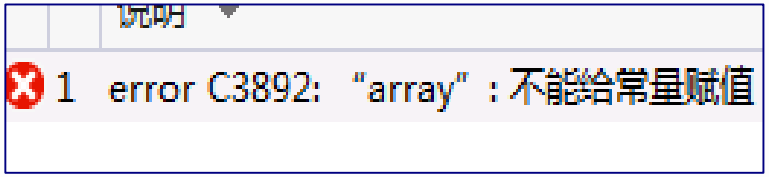
作为

指针变量来处理！

如何“限制”指针实参的功能

指向符号常量的指针

```
#include<iostream>
using namespace std;
int sum(const int array[], int n)
{
    for (int i = 0; i < 10 - 1; i++)
    {
        *(array + 1) = *array + *(array + 1);
        array++;
    }
    return *array;
}
int main()
{
    int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    cout << sum(a, 10);
    return 0;
}
```



符号常量

- 符号常量声明语句：
  - 方式一：const 数据类型 常量名=常量值;
  - 方式二：数据类型 const 常量名=常量值;

```
#include<iostream>
using namespace std;
void main()
{
    const float PI = 3.14159f; // float const PI=3.14159f;
    float r;
    cout << "请输入半径r：";
    cin >> r;
    cout << "圆面积为:" << PI*r*r << endl;
}
```

指向符号常量的指针

- 定义语句：const int *p ;

```
#include <iostream>
using namespace std;
int main()
{
    int a = 256;
    int *p = &a;
    *p = 257;
    cout<<*p<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int a = 256;
    const int *p = &a;
    *p = 257; //错误
     cout<<*p<<endl;
    return 0;
}
```

指向符号常量的指针

- 用途：

```
void mystrcpy(char *dest, const char *src)
{ ……}

int main()
{
   char a[20] = “How are you!”;
   char b[20];
   mystrcpy(b,a);
   cout<<b<<endl;
   return 0;
}
```

保证字符串src不被修改！

关于 指向符号常量的指针

```
#include<iostream>
using namespace std;
int main()
{
    const int a = 78;  const int b = 28;  int c = 18;
    const int * pi = &a;
    *pi = 58;                    // (error, *p不能被赋值)
    pi = &b;                     // (可以给pi重新赋值 )
    *pi = 68;                    //(error, *p不能被赋值)
    pi = &c; *pi = 88;           //(error, *p不能被赋值)
    return 0;
}
```

本节内容

- 指针与函数
  - 指针用做函数参数
    - 如何“限制”指针的功能
  - 指针用做函数返回值
    - 静态局部变量

返回指针值的函数

- 函数的返回值可以是多种类型
  - 返回整型数据的函数：

    int max(int x, int y);
  - 返回指针类型数据的函数

    int *function(int x, int y);
    - 函数名字前面表示函数的类型“*”

返回指针值的函数

- 打印出第二行第三列的值

```
#include<iostream>
using namespace std;
void main(){
    int a[4][4]=
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
    12, 13, 14, 15, 16};
    int *p;
    p = get(a, 2, 3);
    cout<<*p<<endl;
}
```

```
int *get(int arr[ ][4],
                    int n, int m)
{
    int *pt;
    pt = *(arr + n - 1) + m-1;
    return(pt);
}
```

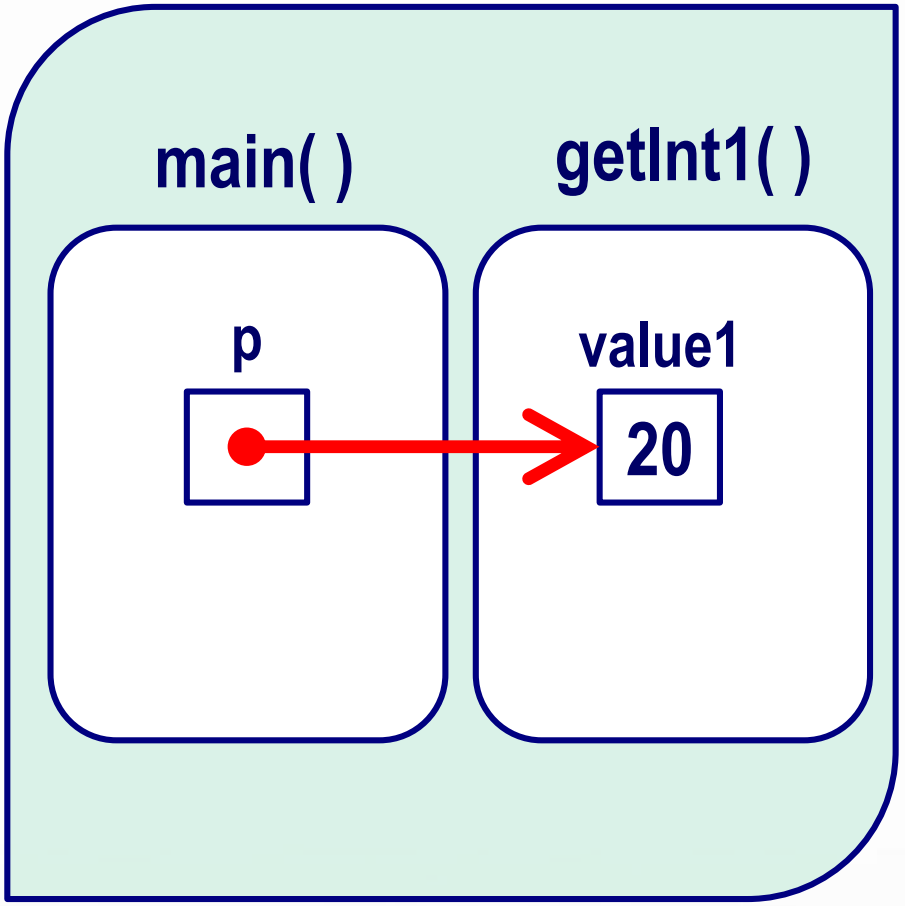
返回指针值的函数

- 判断程序的执行结果：

```
#include<iostream>
using namespace std;
int *getInt1()
{
    int value1 = 20;
    return &value1;
}
int main(){
    int *p;
    p = getInt1();
    cout << *p << endl;
    return 0;
}
```

返回指针值的函数

```
#include<iostream>
using namespace std;
int *getInt1()
{
    int value1 = 20;
    return &value1;
}
int main(){
    int *p;
    p = getInt1();
    cout << *p << endl;
    return 0;
}
```



返回指针值的函数

- 判断程序的执行结果：

```
#include<iostream>
using namespace std;
int main(){
    int *p,*q;
    p = getInt1();
    q = getInt2();
    cout << *p << endl;
    return 0;
}
```

```
int *getInt1()
{
    int value1 = 20;
    return &value1;
}
int *getInt2()
{
    int value2 = 30;
    return &value2;
}
```

确保返回地址的意义

- 返回一个处于生命周期中的变量的地址
  - 返回全局变量的地址，而非局部变量的地址

```
#include<iostream.h>
int value1 = 20;
int value2 = 30;
int main()
{   int *p,*q;
    p = getInt1();
    q = getInt2();
    cout << *p << endl;
    return 0; }
```

```
int *getInt1()
{
        return &value1;
}

int *getInt2()
{
        return &value2;
}
```

确保返回的地址意义

- 返回一个处于生命周期中的变量的地址
  - 返回静态局部变量的地址，而非动态局部变量的地址

```
#include<iostream>
using namespace std;
int main(){
    int *p,*q;
    p = getInt1();
    q = getInt2();
    cout << *p << endl;
    return 0;
}
```

```
int *getInt1()
{
        static int value1 = 20;
        return &value1;
}
int *getInt2()
{
        static int  value2 = 30;
        return &value2;
}
```

什么是静态局部变量

- 静态局部变量
  - 函数中的局部变量的值在函数调用结束后不消失而保留原值
    - 即其占用的存储单元不释放，在下一次该函数调用时，仍可以继续使用该变量；
  - 用关键字static进行声明，可将变量指定为“静态局部变量”。

static int value1 = 20;

```cpp
#include<iostream>
using namespace std;
void function()
{
    int a = 0;
    static int b = 0;
    a = a + 1;
    b = b + 1;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
}
int main()
{
    for (int i = 0; i < 5; i++)
    {
        function();
        cout << "Call Again!" << endl;
    }
    return 0;
}
```

```
a = 1
b = 1
Call Again!
a = 1
b = 2
Call Again!
a = 1
b = 3
Call Again!
a = 1
b = 4
Call Again!
a = 1
b = 5
Call Again!
Press any key to continue
```

静态 vs. 动态

```
#include<iostream>
using namespace std;
void function()
{
    int a = 0;
    static int b = 0;
    a = a + 1;
    b = b + 1;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
}
int main()
{
    for (int i = 0; i < 5; i++)
    {
        function();
        cout << "Call Again!" << endl;
    }
    return 0;
}
```

- 动态变量a的有效范围
- 静态变量b的有效范围

小结

- 指针与函数
  - 指针用做函数参数
    - 函数拿到地址可对其所指内容进行修改；
    - 可以使用const来“限制”指针的功能；
  - 指针用做函数返回值
    - 必须确保函数返回的地址是有意义的；
    - 返回全局变量或静态局部变量；

谢　谢！