



《计算概论A》课程 程序设计部分

C 程序中的函数

李 戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn





本节内容

函数的定义


函数的执行

变量的作用范围

数组与函数

函数举例




$$y = f(x)$$

- 已知一个数，求其平方根

`r = sqrt(100.0);`

- 已知底数 x ，幂指数 y ，求 x^y

`k = pow(x, y);`

- 求一个字符串的长度

`i = strlen (str1);`

- 比较两个字符串的大小

`v = strcmp(str1, str2)`

- 把字符串转换为相应的整数

`n = atoi(str1)`



什么是函数 (1/7)

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

```
int absolute(int n)
{
```

形式参数
(形参)

```
    if (n < 0)
```

```
        return (-n);
```

```
    else
```

```
        return n;
```

```
}
```

```
int main()
{
```

```
    int m = -123, result = 0;
```

```
    result = absolute(m);
```

```
    cout << result;
```

```
    return 0;
```

```
}
```

实际参数
(实参)

什么是函数 (2/7)

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

```
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

```
int main()
{
    int m = 3, n = 4;
    float result = 0;

    result = max(m, n);

    cout << result;
    return 0;
}
```

什么是函数 (3/7)

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

```
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

```
int main()
{
    cout<< max(3, 4);

    return 0;
}
```

什么是函数 (4/7)

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

int get_int( )
{
    int n = 0;
    cout << "Please input an integer: " << endl;
    cin >> n;
    return n;
}

int main()
{
    int result = 0;

    result = get_int();
    cout << result;
    return 0;
}
```

什么是函数 (5/7)

```
#include <iostream>
using namespace std;
void delay(int n)
{
    for (int i = 0; i < n * 100000; i++);
    return;
}
int main()
{
    for (int j = 0; j < 100; j++)
    {
        cout << j << endl;
        delay(1000);
    }
    return 0;
}
```


函数调用的方式

- 以函数在程序中的出现位置和形式来看，函数的调用方式可分为以下3种

① 函数调用**作为独立语句**，例如：

```
stringPrint( );
```

调用函数完成某项功能，没有任何的返回值。

② 函数**作为表达式的一部分**，例如：

```
number = max(numA, numB)/2;
```

③ 以**实参形式**出现在其它函数的调用中。例如：

```
number = min(sum(-5, 100), numC);
```

什么是函数 (6/7)

```
#include <iostream>
using namespace std;
void show( )
{
    cout << "*****" << endl;
    cout << "*   System error has occurred.   *" << endl;
    cout << "* Please contact the administrator. *" << endl;
    cout << "*   Sorry for the inconvenience.   *" << endl;
    cout << "*****" << endl;
}

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

什么是函数 (7/7)

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

    return 0;

}
```

■ 函数是C程序的基本构成单位

- ◆ 一个C程序由一个或多个源程序文件组成。
- ◆ 一个源程序文件可以由一个或多个函数组成。

■ 函数都是有“类型”的

- ◆ 函数的类型是指
“函数的 返回值 的 数据类型”

函数的声明

```
#include <iostream>
using namespace std;
float max(float a, float b)
{
    if (a > b)
        return a;
    else
```

```
#include <iostream>
using namespace std;
float max(float, float);
int main()
```

**函数的原型 =
返回值类型 + 函数名 + 参数类型**

```
{
    cout<< max(3, 4);
    return 0;
}
```

```
{
    cout<< max(3, 4);
    return 0;
}
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

函数的原型 及 函数声明

■ 函数原型

- ◆ 由函数的返回类型、函数名以及参数表构成的一个符号串，其中参数可不写名字。

`bool checkPrime (int)`

- ◆ 函数的原型又称为函数的 “Signature”

■ 函数的声明

- ◆ 函数在使用前都要声明，除非被调用函数的定义部分已经出现在主调函数之前。
- ◆ 在C语言中，函数声明就是函数原型。

函数的声明

max.h

```
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

compare.cpp

```
#include <iostream>
#include "max.h"
using namespace std;
int main()
{
    cout<< bigger(3, 4);
    return 0;
}
```



本节内容

函数的定义

函数的执行

变量的作用范围

数组与函数

函数举例



函数的执行

```
#include <iostream>
using namespace std;
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
int main()
{
    int m = 3, n = 4;
    float result = 0;
    result = max(m, n);
    cout << result;
    return 0;
}
```

main()

```
int main()
{
    int m = 3, n = 4;

    float result = 0;

    result = max(m, n);

    cout << result;

    return 0;
}
```

max()

```
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```


函数的执行

```
#include <iostream>
using namespace std;
float max(float a, float b)
{
```

- (1) 初始化max();
- (2) 传递参数;
- (3) 保存当前现场;

```
    return b;
```

```
}
```

```
int main()
```

- (1) 接收函数的返回值;
- (2) 恢复现场, 从断点处继续执行;

```
    return 0;
```

```
return 0;
```

```
}
```

main()

```
int main()
{
    int m = 3, n = 4;

    float result = 0;

    result = max(m, n);

    cout << result;

    return 0;
}
```

max()

```
float max(float a, float b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

函数参数的传递

```
int main( )
```

```
{
```

```
float result = 0;
```

```
result = max(3, 4);
```

```
cout << result;
```

```
return 0;
```

```
}
```

```
float max(float a, float b)
```

```
{
```

```
if (a > b)
```

```
return a;
```

```
else
```

```
return b;
```

```
}
```

函数参数的传递

COPY!

```
int main( )
```

```
{
```

```
    int m = 3, n = 4;
```

```
    float result = 0;
```

```
    result = max(m, n);
```

```
    cout << result;
```

```
    return 0;
```

```
}
```

```
float max(float a, float b)
```

```
{
```

```
    if (a > b)
```

```
        return a;
```

```
    else
```

```
        return b;
```

```
}
```



函数参数的传递

■ 参数的传递

- ◆ 实参与形参具有不同的存储单元，实参与形参变量的数据传递是“值传递”；
- ◆ 函数调用时，系统给形参分配存储单元，并将实参对应的值传递给形参；

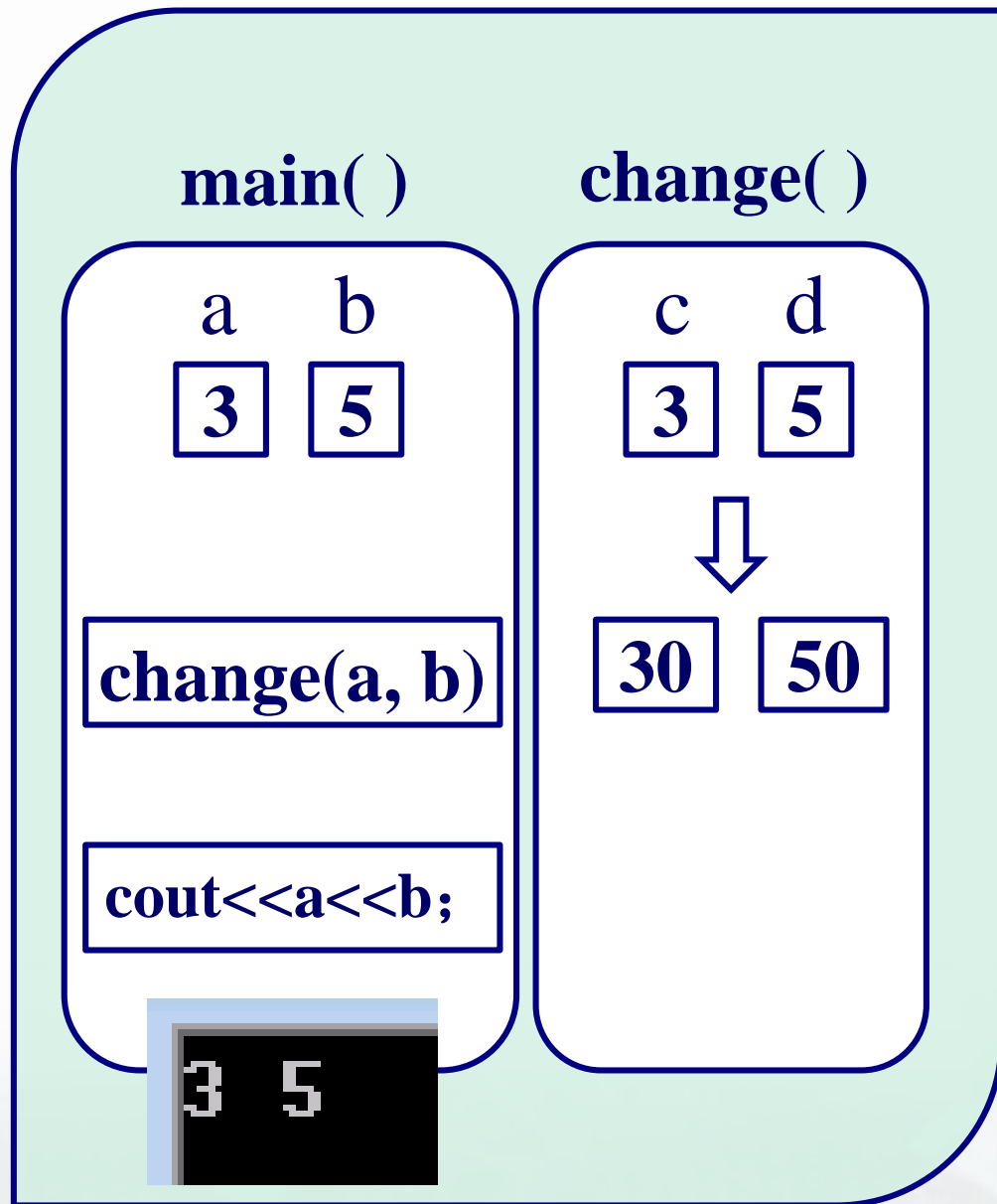
P.S. 实参与形参的类型必须相同或可以兼容；



几道现场思考题

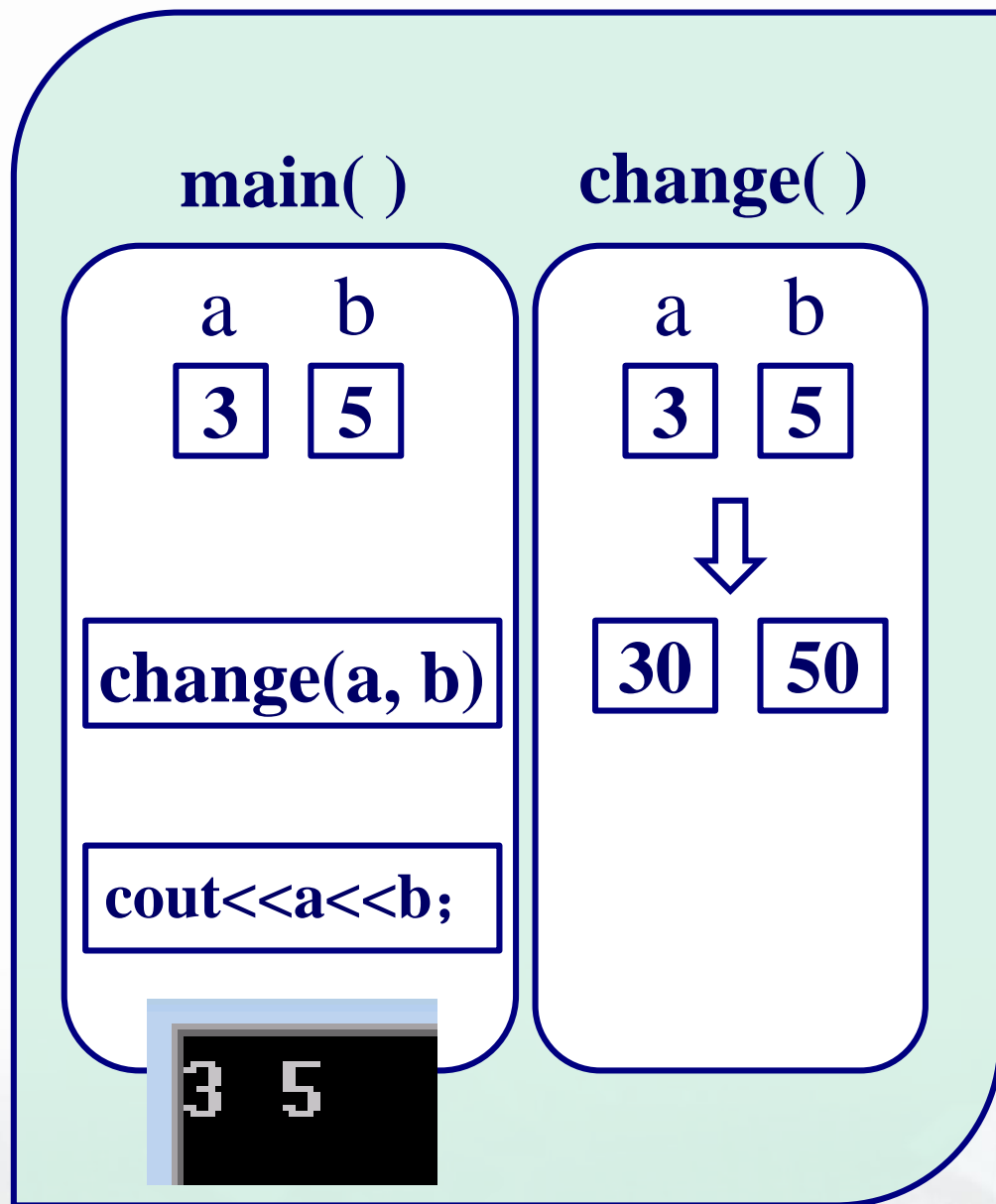
现场思考题 (1)

```
#include <iostream>
using namespace std;
void change(int c, int d)
{
    c = 30; d = 50;
}
int main()
{
    int a = 3, b = 5;
    change(a, b);
    cout<<a<<" "<<b;
    return 0;
}
```



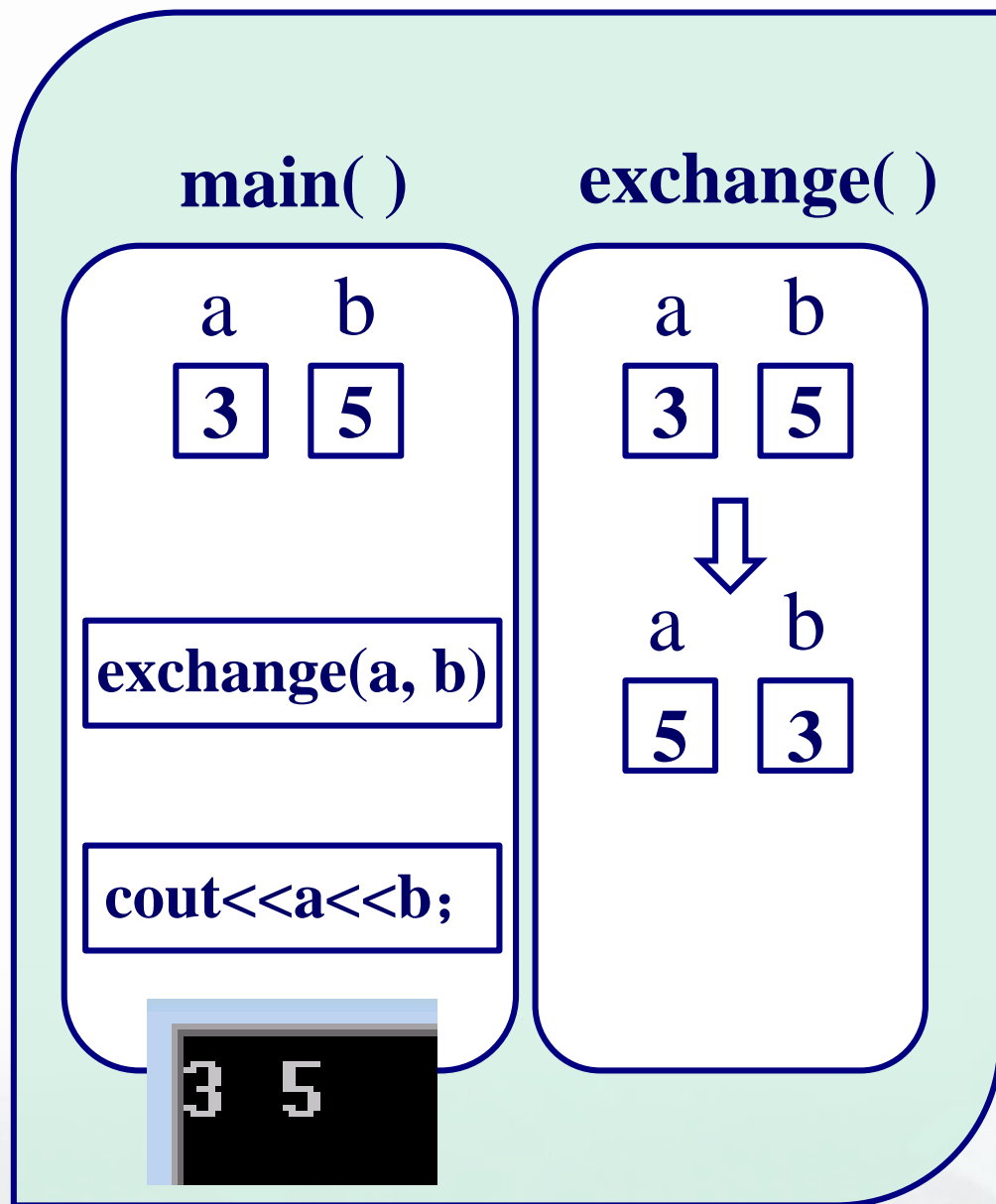
现场思考题 (2)

```
#include <iostream>
using namespace std;
void change(int a, int b)
{
    a = 30; b = 50;
}
int main()
{
    int a = 3, b = 5;
    change(a, b);
    cout<<a<<" "<<b;
    return 0;
}
```



现场思考题 (3)

```
#include<iostream>
using namespace std;
void exchange(int a, int b)
{
    int p;
    if (a < b)
    {
        p = a; a = b; b = p;
    }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<< endl;
    return 0;
}
```





本节内容

函数的定义

函数的执行

变量的作用范围

数组与函数

函数举例



局部变量与全局变量

- 根据变量在程序中作用范围的不同，可以将变量分为：

- ◆ **局部变量**

在函数内或块内定义，只在这个函数或块内起作用的变量；

- ◆ **全局变量**

在所有函数外定义的变量，它的作用域是从定义变量的位置开始到本程序文件结束。

刚才的例子

```
#include<iostream>
using namespace std;
void exchange(int a, int b)
{
    int p;
    if (a < b)
    {
        p = a; a = b; b = p;
    }
}
```

a, b 的势力范围

形参是局部变量

```
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<< endl;
    return 0;
}
```

a, b 的势力范围

全局变量

```
#include<iostream>
```

```
using namespace std;
```

```
int excel_number = 0;
```

```
void excel_count( float score )
```

```
{
```

```
    if (score > 85)
```

```
    {
```

```
        excel_number++;
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    float score = 0;
```

```
    for (int i = 0; i < 100; i++)
```

```
    {
```

```
        cin >> score;
```

```
        excel_count(score);
```

```
    }
```

```
    cout << excel_number << endl;
```

```
    return 0;
```

```
}
```

局部变量

score
的作用范围

全局变量

excel_number
的作用范围

局部变量

i
的作用范围

局部变量

score
的作用范围

局部 vs. 全局

```
#include<iostream>
using namespace std;
int a = 0, b = 0;
void exchange( )
{
    int p;
    if (a < b)
    {
        p = a; a = b; b = p;
    }
}
int main()
{
    cin >> a >> b;
    exchange( );
    cout<<a<<" "<<b<< endl;
    return 0;
}
```

a b
[0] [0]



a b
[5] [3]

main()

cin>>a>>b;

exchange(a, b)

cout<<a<<b;

exchange()

p = a;
a = b;
b = p;

局部 vs. 全局

```
#include<iostream>
using namespace std;
int a = 0, b = 0;
void exchange(int a, int b)
{
    int p;
    if (a < b)
    {
        p = a; a = b; b = p;
    }
}
int main()
{
    cin>>a>>b;
    exchange(a, b);
    cout<<a<<" "<<b<< endl;
    return 0;
}
```

局部变量
a, b
的作用范围

全局变量
a, b
的作用范围

- 当全局变量与局部变量同名时，局部变量将在自己作用域内有效，它将屏蔽同名的全局变量。

局部 vs. 全局

```
#include<iostream>
using namespace std;
int a = 0, b = 0;
void exchange(int a, int b)
{
    int p;
    if (a < b)
    {
        p = a; a = b; b = p;
    }
}
int main()
{
    cin>>a>>b;
    exchange(a, b);
    cout<<a<<" "<<b<< endl;
    return 0;
}
```

a b

0	0
---	---

main()

cin>>a>>b;

exchange(a, b)

cout<<a<<b;

exchange()

a b

3 5

3 5

5

3



本节内容

函数的定义

函数的执行

变量的作用范围

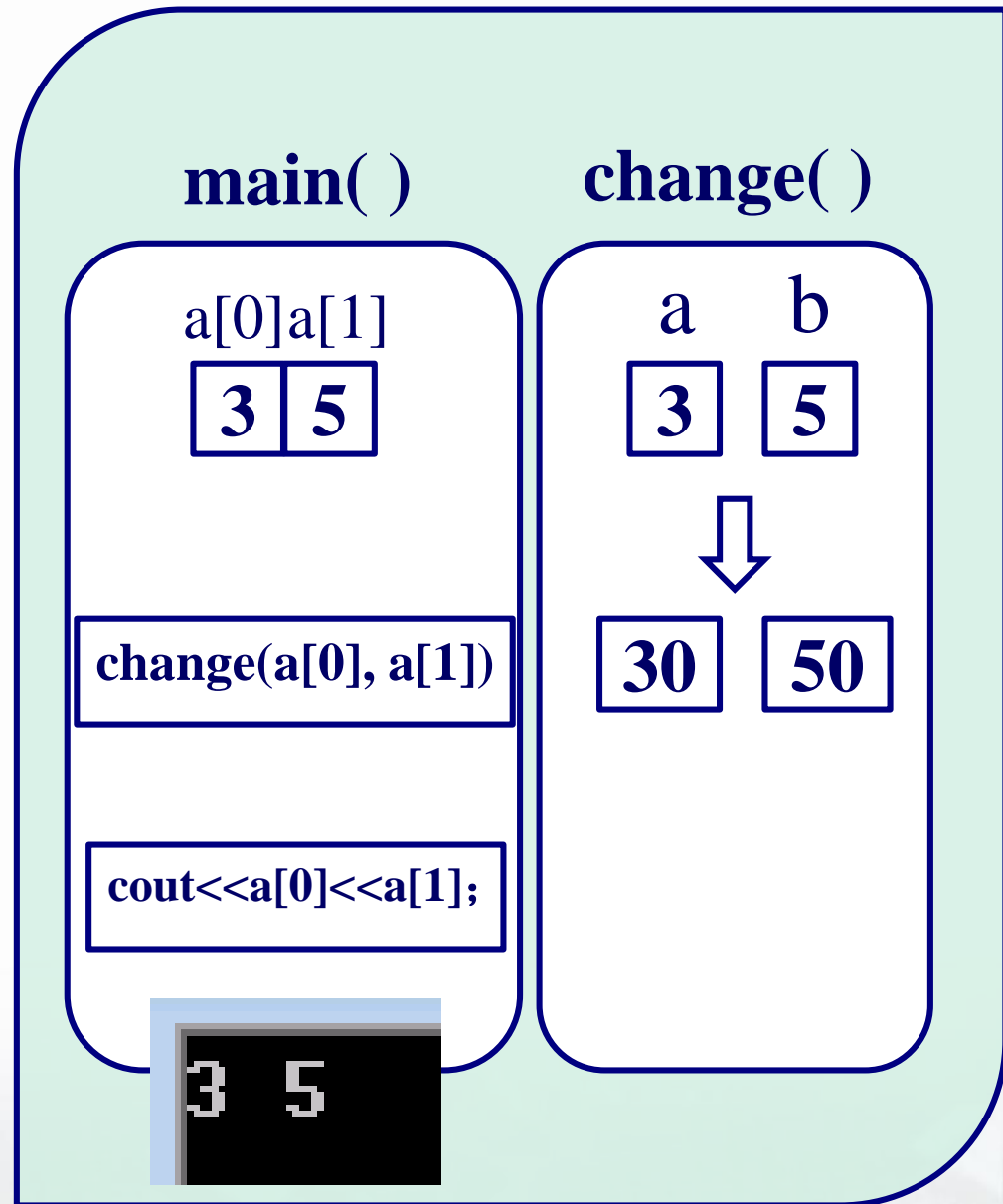
数组与函数

函数举例



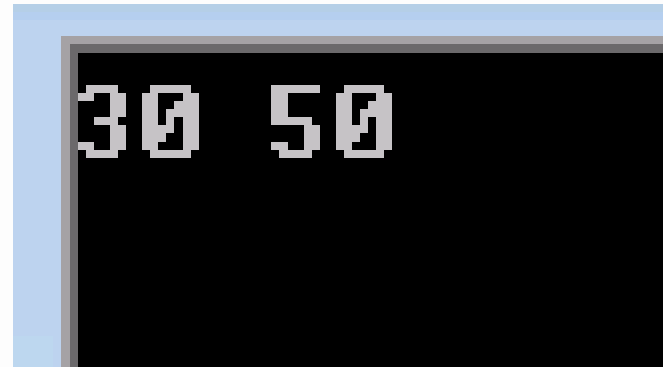
数组元素做函数参数

```
#include <iostream>
using namespace std;
void change(int a, int b)
{
    a = 30; b = 50;
}
int main()
{
    int a[2] = {3, 5};
    change(a[0], a[1]);
    cout<<a[0]<<" "
        <<a[1]<<endl;
    return 0;
}
```



数组元素做函数参数

```
#include <iostream>
using namespace std;
void change(int a[])
{
    a[0] = 30; a[1] = 50;
}
int main()
{
    int a[2] = { 3, 5 };
    change(a);
    cout << a[0] << " "
         << a[1] << endl;
    return 0;
}
```



30 50

数组名 做函数参数

COPY!

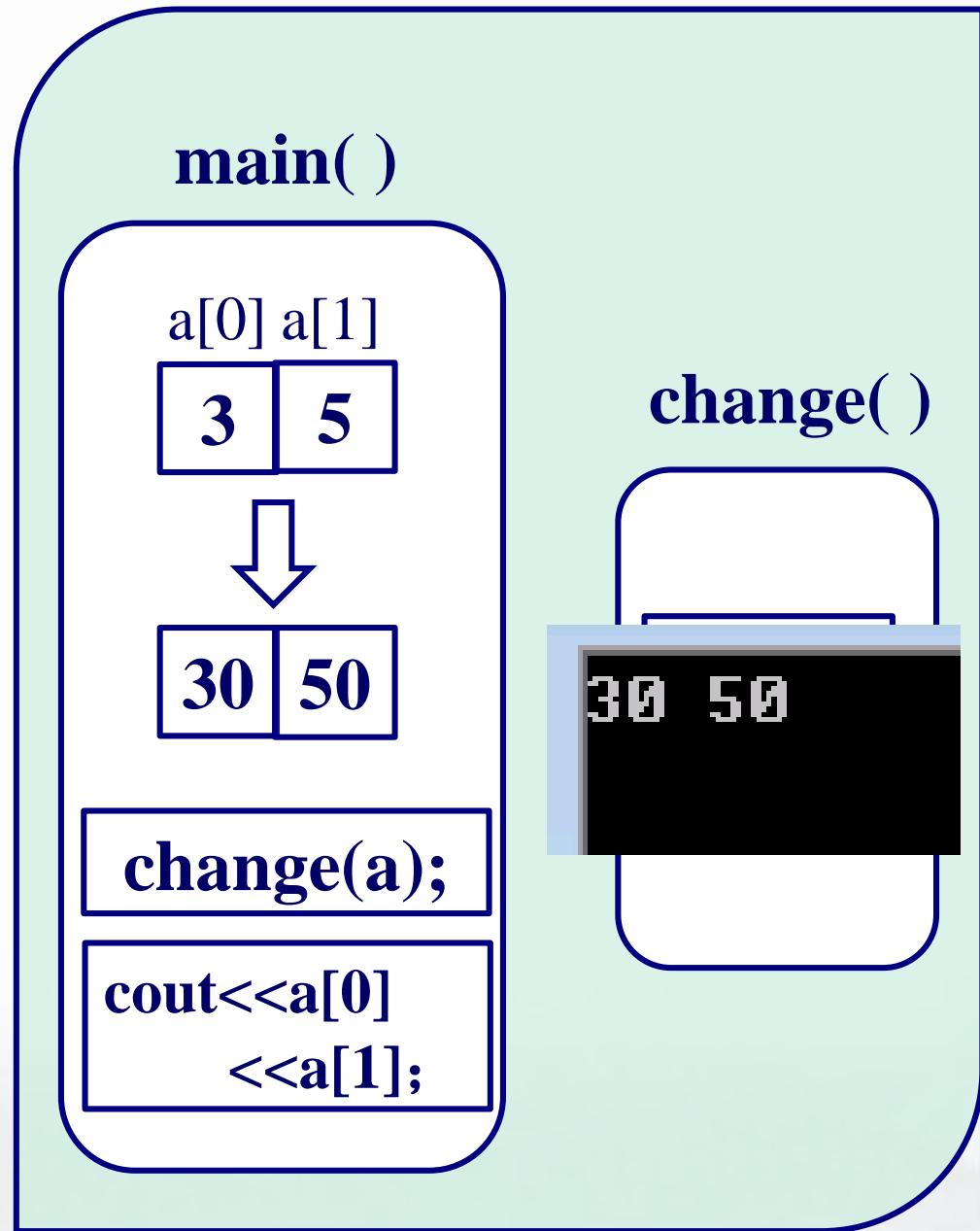
```
#include <iostream>
using namespace std;
int main()
{
    int a[2] = { 3, 5 };
    change(a);
    cout << a[0] << " "
         << a[1] << endl;
    return 0;
}
```

```
void change(int a[])
{
    a[0] = 30; a[1] = 50;
}
```

**a , 不是变量 !
是数组在内存中的地址 !**

数组名 做函数参数

```
#include <iostream>
using namespace std;
void change(int a[])
{
    a[0] = 30; a[1] = 50;
}
int main()
{
    int a[2] = { 3, 5 };
    change(a);
    cout << a[0] << " "
         << a[1] << endl;
    return 0;
}
```

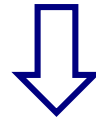


数组名 做函数参数

```
#include <iostream>
using namespace std;
void change(int a[])
{
    a[0] = 30; a[1] = 50;
}
int main()
{
    int a[2] = { 3, 5 };
    change(a);
    cout << a[0] << " "
         << a[1] << endl;
    return 0;
}
```

main()

a[0] a[1]



change(a);

cout<<a[0]
<<a[1];

change()

int a[];



本节内容

函数的定义

函数的执行

变量的作用范围

函数的作用





日历问题

■ 问题描述

- ◆ 给定从公元2000 年1月1日开始逝去的天数，请编写程序给出这一天是哪年哪月哪日星期几。
 - 注意闰年：闰年被定义为能被4整除的年份，但是能被100整除而不能被400 整除的年是例外，它们不是闰年。
 - ◆ 例如：1700, 1800, 1900和2100不是闰年，而1600, 2000 和 2400是闰年。
 - 注意每个月的天数不一样！

日历问题

■ 输入输出要求

- ◆ 输入多组数据，每组一个正整数，表示从2000年1月1日开始已经过去的天数。
- ◆ 对输入的每个天数，输出一行，该行包含对应的日期和星期几。格式为：

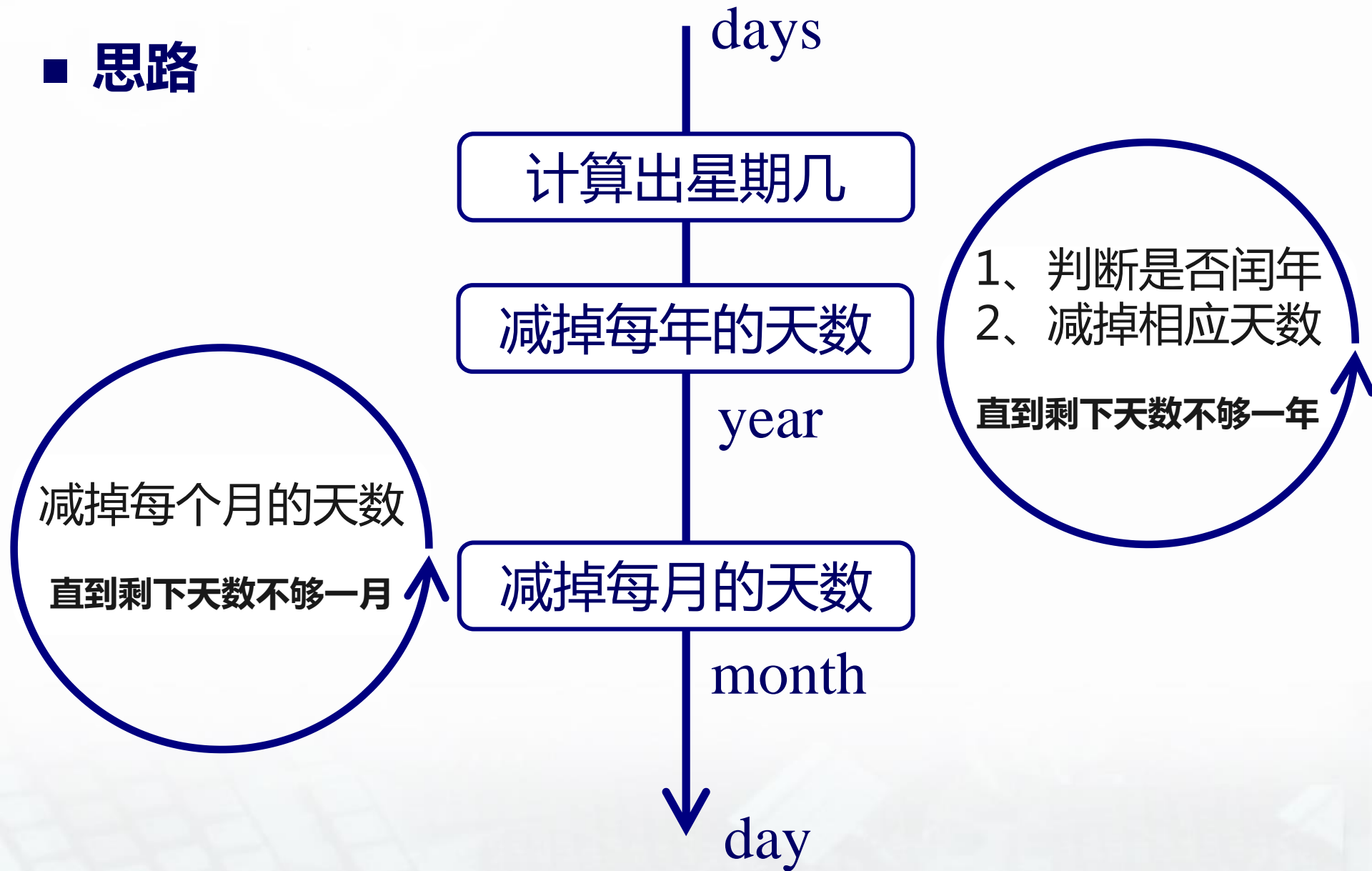
"YYYY-MM-DD DayOfWeek"

其中 "DayOfWeek" 必须是：Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday。

- ◆ 输入最后一行是-1，不必处理。可以假设结果的年份不会超过9999。

日历问题

■ 思路





日历问题

(1) 计算星期几

```
int get_dayofweek( )  
{  
    int dayofweek;  
    dayofweek = days % 7;  
    return dayofweek;  
}  
  
char week[7][10] = {“Saturday”, “Sunday”, “Monday”,  
“Tuesday”, “Wednesday”, “Thursday”, “Friday”};
```

日历问题

(2) 计算年数

```
int get_year( )
{
    int i = 2000, leap_year;
    while(1){
        leap_year = (i % 4 == 0 && i % 100 != 0 || i % 400 == 0);
        if(leap_year==1&&days>=366)
        { days = days - 366;    i++; continue;}
        else if(leap_year==0&&days>=365)
        {days = days - 365;    i++; continue;}
        else
            break;
    }
    return i;
}
```

日历问题

(3) 计算月份

```
int get_month(int leap_year)
{
    int pmonth[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int rmonth[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int j = 0;
    while(1){
        if(leap_year==1 && days>=rmonth[j])
        {
            days=days-rmonth[j];
            j++;
        }
        else if(leap_year==0 && days>=pmonth[j])
        {
            days = days-pmonth[j];
            j++;
        }
        else break;
    }
    return ++j;
}
```

```
#include<iostream>
using namespace std;
int days;
int get_dayofweek();
int get_year();
int get_month(int);
int main()
{
    int year, month, dayofweek; int leap_year;
    char week[7][10] = {"Saturday", "Sunday", "Monday", "Tuesday",
        "Wednesday", "Thursday", "Friday"};
    while ((cin>>days)&& days != -1) {
        dayofweek=get_dayofweek();
        year = get_year();
        leap_year = (year%4==0&&year%100!=0||year%400==0);
        month = get_month(leap_year);
        cout<<year<<"-"<<month<<"-"<<++days<<" "<<week[dayofweek];
    }
    return 0;
}
```



小结

■ 全局变量

- ◆ 破坏了函数的“相对独立性”；
- ◆ 增加了函数之间的“耦合性”；
- ◆ 函数之间的交互不够清晰；

■ 因此：

- ◆ 不在非常必要的情况下，不要使用全局变量。



谢谢！