程序设计基础

教学团队：徐明星，兴军亮，任炬

renju@tsinghua.edu.cn

2024秋，每周一第2节，三教2301

疑问解答与作业回顾

【练习题01】 第5题（填空题）

以下程序运行的结果是 [填空1]

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

int main()
{
    int a = 123;
    int *ptr = &a;

    *ptr = 974;
    cout << "a = " << a << endl;

    return 0;
}
```

答案：a = 974

【练习题01】 第9题（填空题）

运行以下程序，输入字符N，
程序的输出结果为 [填空1]

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

int main()
{
    char in;
    cin >> in;

    if (in = 'Y')
        cout << "YES";
    else
        cout << "NO";

    return 0;
}
```

答案：YES

【练习题02】 第1题（单选题）

以下代码片段的输出结果为

```
for (int i=0; i<4; i++);
      cout << "ABC";
```

A ABC

B ABCABCABCABC

C ABCABCABC

D ABC ABC ABC

答案：A

【练习题03】 第3题（不定项选择题）

关于符号*，以下哪些说法是正确的：

A 既可以用于定义指针变量的语句，也可以用于使用指针变量的语句

B 可以出现在算术运算表达式中

C 用在指针变量前，可以得到该指针变量所在内存的地址

D 用在指针变量前，表示读取指针变量自己内存单元的内容

答案：AB

【练习题03】 第5题（不定项选择题）

对以下代码片段，哪些说法错误?

```
int a, b;
int add(int a, b)
{
    return a + b;
}
```

A 函数形式参数不能与全局变量重名

B 有语法错误，编译不通过

C 表达式a+b的值应先赋给一个局部变量，才能用return语句返回

D 调用add时，必须有一个变量来保存它的返回值

答案：**ACD**

【练习题03】 第8题（不定项选择题）

与以下代码等效的选项是：

```
if (C1 || C2) {
    if (C3) S1;
}
else S2;
```

其中，C1和C2表示条件表达式，S1和S2表示语句

A
```
if (C1 || C2 && C3) S1;
else S2;
```

B
```
if (C1 && C3 || C2 && C3) S1;
else S2;
```

C
```
if (!C1 && !C2) S2;
else if (C3) S1;
```

D
```
if (C1 && C3) S1;
if (C2 && C3) S1;
if (!C1 && !C2) S2;
```

答案：C

【练习题03】 第8题（不定项选择题）

优先级		操作符	描述	运算顺序
1	作用域	::	作用域	从左到右
2	后缀（单目运算符）	++ --	增量或减量	
		()	函数形式	
		[]	下标	
		. ->	获取成员	
3	后缀（单目运算符）	++ --	增量或减量	从右到左
		~ !	按位取反或非	
		+ -		
		& *	取地址/取值	
		new delete	动态分配与解除分配	
		sizeof	取内存大小	
		(*type*)	强制类型转换	
4	指向成员	.* ->*	指向符号	从左到右
5	乘、除、取余	* / %	乘、除、取余	
6	加、减	+ -	加、减	
7	按位左移、右移	<< >>	按位左移、右移	
8	比较运算符	< > <= >=	比较运算符	
9	相等号不等号	== !=	相等号不等号	
10	按位与	&	按位与	
11	按位异或	^	按位异或	
12	按位或	|	按位或	
13	逻辑与	&&	逻辑与	
14	逻辑或	||	逻辑或	
15	赋值表达式	= *= /= %= += -= >>= <<= &= ^= |=	复合运算符	从右到左
		?:	三目运算符	
16	排序	,	逗号运算符	从左到右

【练习题03】 第8题（不定项选择题）

C1	C2	C3	原表达式执行语句	A	B	C	D
0	0	0	S2	S2	S2	S2	S2
0	0	非0	S2	S2	S2	S2	S2
0	1	0	不执行	S2	S2	不执行	不执行
0	1	非0	S1	S1	S1	S1	S1
1	0	0	不执行	S1	S2	不执行	不执行
1	0	非0	S1	S1	S1	S1	S1
1	1	0	不执行	S1	S2	不执行	不执行
1	1	非0	S1	S1	S1	S1	S1S1

第5讲 数据组织与处理（上）

【任务5.1】 歌手比赛评分系统

某电视台举办歌手大奖赛，邀请了7名评委给歌手打分。在计算歌手的平均得分时，为了公平起见，要把这些评委给出的最高分、最低分都去掉，只将剩下的5名评委给出的评分进行平均。

为了提高效率和保存记录，在评委报出成绩后，工作人员将分数录入计算机中，通过专门编写的程序来计算和输出歌手的最终得分。

请你编写程序来完成这个任务。设评委给分区间为0~100分。

呃？ 7个变量！怎么找最大、最小值？

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

int main() {
    int sum = 0;
    int min = 100, max = 0;
    int j1, j2, j3, j4, j5, j6, j7;

    ???

    return 0;
}
```

【编程技巧】一次性流水作业

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

int main() {
    int sum = 0;
    int min = 100, max = 0;
    for (int i = 0; i < 7; i++) {
        int score;
        cin >> score;
        if (score > max) max = score;
        if (score < min) min = score;
        sum += score;
    }
    cout << "Score: " << (sum - max - min) / 5.0 << endl;
    return 0;
}
```

“雁过拔毛”法

这两个条件互相独立
不应使用“IF-ELSE”
先后次序也无关紧要

需使用浮点类型

如果还要算7个数的方差呢？有更好的方法吗？

需要一种定义带下标的相互关联的一批变量的办法！

数组：带下标的相互关联的变量序列

数组是计算机语言提供的组织多个数据的一种重要方式：

- 提供了多个同类型的数据（值）在内存中连续存放的工具
- 提供了对大量内存单元进行高效“命名”的途径
- 提供了在程序运行过程中动态改变“变量名称”的手段
- 是一些重要算法思想的实现基础

数组定义的语法：如何定义数组？

类型说明符 数组名[常量表达式];

TYPE array_name[const_expr];

例：

```
float  sheep_weight[10];
int    __a2001[1000];
char   student_name[20];
```

说明：

1. 数组变量的名称，必须符合语言对变量命名的要求；
2. 用方括号将常量表达式括起；
3. 常量表达式定义了数组元素的个数；

如果使用变量来定义数组，在使用数组前，变量要确定数值。比如：

int len; cin<<len; int num[len]; (合法性依赖于编译器)

C/C++语言中的各种“括号”

- 表达式调整计算优先次序：使用圆括号
- 函数调用：使用圆括号
- 函数定义：使用花括号
- 头文件：使用尖括号
- 语句块（IF-ELSE，FOR）：使用花括号
- 数组：使用方括号

https://zhuanlan.zhihu.com/p/137564717

数组每个元素都可视为“变量”

数组中每个元素所在的内存单元，可以通过“数组名[位置下标]”来访问（赋值、读取）。

在C/C++语言中，数组元素的位置下标从 0 开始计数（编号）。

例如，int a[5]；定义了一个含有5个整数的数组，各元素的“变量名称”分别为：

a[0], a[1], a[2], a[3], a[4]

是5个带下标的变量，它们的类型是相同的。定义上述数组的效果与下面的变量定义相同：

int a0, a1, a2, a3, a4;

数组变量的初始化

所有类型的数组均可以用下面的格式来进行初始化：

```
type_name array_name[N] = { v1, v2, ... , vN };
```

其中，N 是数组大小，即元素个数，v1, v2, ... , vN 等表示常量表达式。

定义数组并同时进行初始化时，N可以省略，编译器会自动补上。例如：

```
int a[] = { 3, 5, 4, 1, 2 }; /// 等同于int a[5]={3,5,4,1,2};
char b[] = {'c', 'h', 'i', 'n', 'a' };
double c[] = {0.1, 0.2}; /// 等同于 double c[2] = {0.1, 0.2};
```

如果初始化值个数比数组的元素个数要少，则没有对应初始值的元素都被编译器自动设置为0。 例如：

```
int prime[101] = {1, 1};  // 则prime[2], prime[3], .... 等均为0
```

数组变量的初始化

如果是由字符（char）组成的数组，则还可以使用下面更便捷的初始化形式：

```
char array_name[N] = "各种字符"; //字符个数最多等于N-1
```

例如：

```
char b[6] = "china";
```

注意：对字符数组的定义，N必须是字符个数+1，上面语句的初始化结果等效于：

```
char b[6];
b[0] = 'c'; b[1] = 'h'; b[2] = 'i';
b[3] = 'n'; b[4] = 'a'; b[5] = '\0';   // 或 b[5] = 0;
```

或

```
char b[6] = {'c', 'h', 'i', 'n', 'a'}; // 编译器自动将最后元素置0
```

或

```
char b[6] = {'c', 'h', 'i', 'n', 'a', '\0'};
```

或

```
char b[6] = {'c', 'h', 'i', 'n', 'a', 0};
```

投票(匿名) 最多可选1项

设置

【问卷调查】以下代码编译时会发生什么?

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

int main()
{
   char b[4] = "china";
   cout << b << endl;
   return 0;
}
```

A 只有编译警告

B 会出编译错误

C 编译和运行都会一切正常

D 编译正常，运行崩溃

E 我不知道，需要上机试一下

提交

VS2012 开发人员命令提示

```
d:\>type L05-01.cpp
#include <iostream>
using namespace std;

int main() {
        char b[4] = "china";
        cout << b << endl;
        return 0;
}

d:\>g++ L05-01.cpp
L05-01.cpp: In function 'int main()':
L05-01.cpp:5:14: error: initializer-string for array of chars is too long [-fpermissive]
  char b[4] = "china";
              ^~~~~~~

d:\>cl L05-01.cpp
用于 x86 的 Microsoft (R) C/C++ 优化编译器 17.00.61030 版版权所有(C) Microsoft Corporation。
保留所有权利。

L05-01.cpp
C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\INCLUDE\xlocale(336) : warning C4530:
 使用了 C++ 异常处理程序，但未启用展开语义。请指定 /EHsc
L05-01.cpp(5) : error C2117: “b”: 数组界限溢出
        L05-01.cpp(5) : 参见“b”的声明
        L05-01.cpp(5) : 参见“b”的声明

d:\>cl L05-01.cpp /EHsc
用于 x86 的 Microsoft (R) C/C++ 优化编译器 17.00.61030 版版权所有(C) Microsoft Corporation。
保留所有权利。

L05-01.cpp
L05-01.cpp(5) : error C2117: “b”: 数组界限溢出
        L05-01.cpp(5) : 参见“b”的声明
        L05-01.cpp(5) : 参见“b”的声明
```

数组变量的赋值

```
int a[5];
char b[5];
```

先定义再赋值

```
a = {1, 2, 3, 4, 5};
b = "CHINA";
```

所有类型的数组变量**都**不可以直接赋值！

如何给数组中的（各）元素赋值？

- 方法1：一个一个地，单独给指定位置的元素赋值

```
int a[5];
a[0] = 1;
a[3] = 23;
```

- 方法2：使用for语句，成批地连续赋值（逐一赋值）

```
char a[5];
for (int i = 0; i < 5; i++)
     a[i] = 'A' + i;
```

如何给数组中的（各）元素赋值？

- 方法3：通过cin操作，对字符数组变量进行“赋值”

```
#include <iostream>
int main() {
    char h[] = "123456789"; // 这是初始化，不是赋值
    std::cin >> h; // 这是在赋值！
    std::cout << h << std::endl;
    return 0;
}
```

方法3、4仅对字符数组有效

- 方法4：通过strcat, strcpy等函数调用，对字符数组变量进行“赋值”

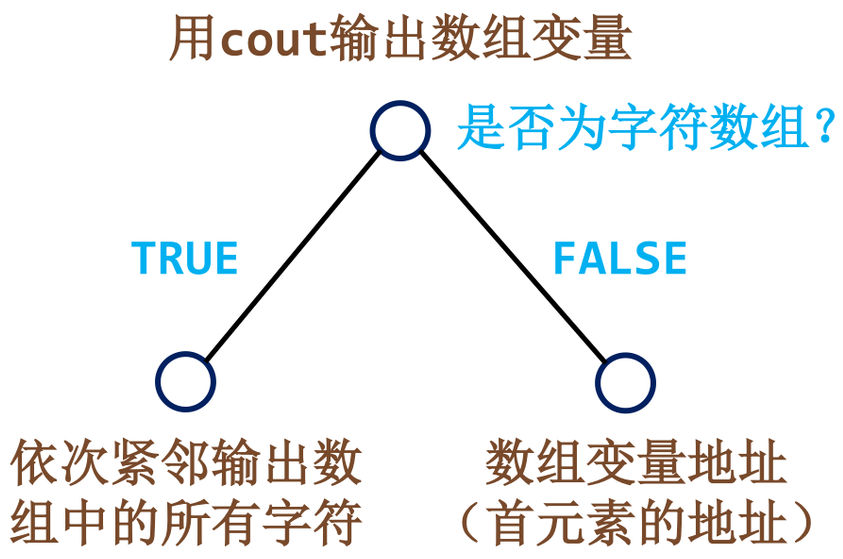
```
#include <iostream>
#include <cstring>
int main() {
    char h[] = "123456789";
    strcpy(h, "FOP22");
    std::cout << h << std::endl;
    return 0;
}
```

```
#include <iostream>
#include <cstring>
int main() {
    char h[30] = "123456789";
    strcat(h, "FOP22");
    std::cout << h << std::endl; //123456789FOP22
    return 0;
}
```

数组变量的输出（cout）

（1）如果是字符数组变量，则cout会将数组的所有元素一齐输出出来*，字符之间无空格。

（2）如果是其他类型的元素组成的数组（变量），则cout只会输出该数组变量的地址，也即数组第一个元素所在内存单元的地址。



cout是如何根据数据类型来决定输出方式的呢？这个问题将在下学期《面向对象程序设计（C++）》中予以解答。

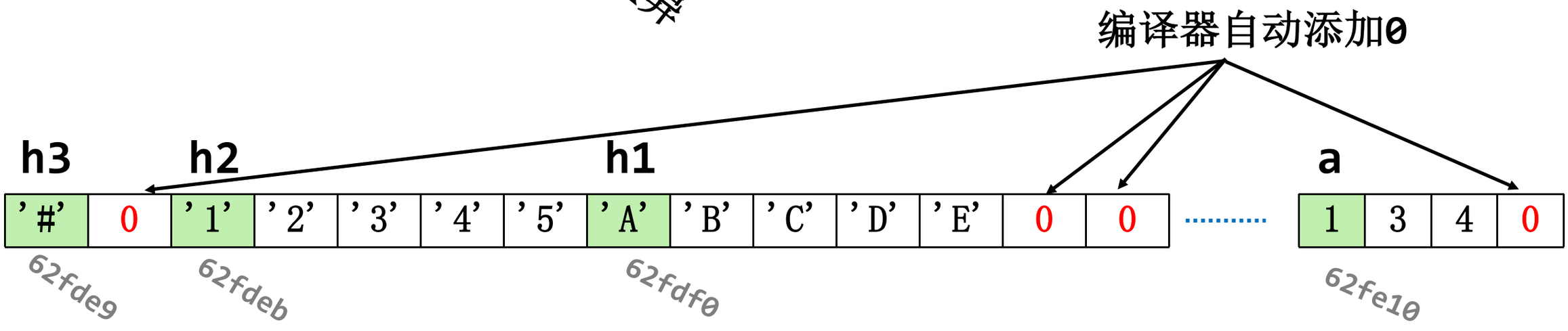
数组变量的输出（cout）

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

int main() {
    int a[4] = {1, 3, 4};
    char h1[30] = "ABCDE",
         h2[5] = {'1', '2', '3', '4', '5'},
         h3[2] = {'#'};
    cout << a << ' ' << h1 << ' ' << h2 << ' ' << h3 << ' ' << int(h3[1]) << endl;
    cout << &a[0] << ' ' << (int*)h1 << ' ' << (int*)h2 << ' ' << (int*)h3 << endl;
    return 0;
}
```

注意输出格式差异

程序运行输出（WIN 11，VS CODE， g++）

```
0x62fe10 ABCDE 12345ABCDE # 0
0x62fe10 0x62fdf0 0x62fdeb 0x62fde9
```

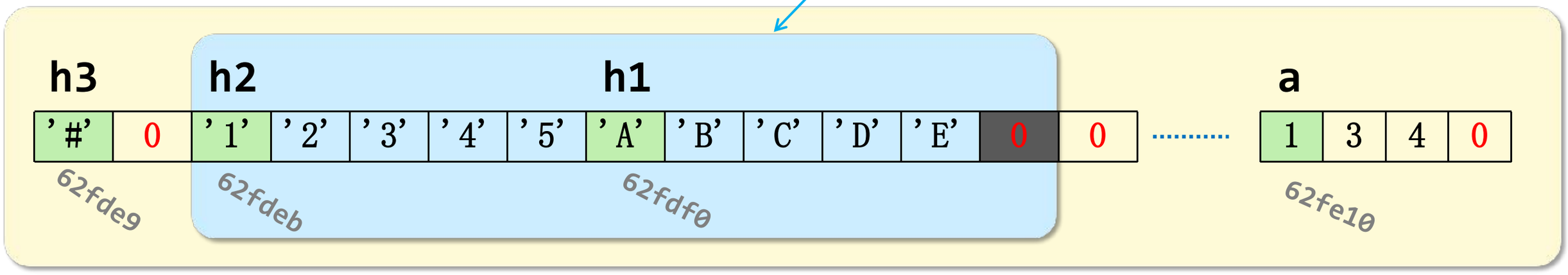


输出字符指针、字符数组变量的内容（cout，printf）

程序运行输出（WIN 11，VS CODE， g++）

```
0x62fe10 ABCDE 12345ABCDE # 0
0x62fe10 0x62fdf0 0x62fdeb 0x62fde9
```

对于语句 cout << h2;
由于h2数组不是以0结束的，所以输出会一直进行到h1所在内存空间中的0处才停止。



数组变量的输出（cout）

（1）如果是字符数组变量，则cout会将数组的所有元素一齐输出出来*，字符之间无空格。

对字符指针、字符数组变量，用**cout**输出它时，将从首地址单元内容开始依次向后输出，一直到某个内存单元的内容是**'\0'**（即**0**）才会停止。

【探究验证】数组大小与元素大小

```
sizeof(int[10])    = 40, sizeof(int)*10    = 40
sizeof(char[10])   = 10, sizeof(char)*10   = 10
sizeof(double[10]) = 80, sizeof(double)*10 = 80
```

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

int main()
{
    int array[10];
    char message[10];
    double salary[10];

    cout << "sizeof(int[10])    = " << sizeof(array)
         << ", sizeof(int)*10    = " << sizeof(int) * 10 << endl;
    cout << "sizeof(char[10])   = " << sizeof(message)
         << ", sizeof(char)*10   = " << sizeof(char) * 10 << endl;
    cout << "sizeof(double[10]) = " << sizeof(salary)
         << ", sizeof(double)*10 = " << sizeof(double) * 10 << endl;

    return 0;
}
```

【探究验证】指针与数组基本等价

```
ptr   = 0x62fde0, ptr + 1   = 0x62fde4
array = 0x62fde0, array + 1 = 0x62fde4
array[0]:       0 0 0 0
array[1]:       2 2 2 2
array[2]:       4 4 4 4
array[3]:       6 6 6 6
array[4]:       8 8 8 8
array[5]:       10 10 10 10
array[6]:       12 12 12 12
array[7]:       14 14 14 14
array[8]:       16 16 16 16
array[9]:       18 18 18 18
```

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

int main() {
    int array[10], *ptr;
    ptr = array;
    cout << "ptr   = " << ptr   << ", ptr + 1   = " << ptr + 1   << endl;
    cout << "array = " << array << ", array + 1 = " << array + 1 << endl;

    for (int i = 0; i < 10; i++)
        if (i % 2 == 0) *(array + i) = i * 2;
        else ptr[i] = i * 2;

    for (int i = 0; i < 10; i++)
        cout << "array[" << i << "]:\t"
             << ptr[i] << ' ' << *(ptr + i) << ' '
             << array[i] << ' ' << *(array + i) << endl;

    return 0;
}
```

整数占4字节，指针变量+1，其指针值（地址值）+4

探究验证时，注意输出必要提示信息，注意**代码排版**，注意输出**内容排版**。

【任务5.1】 基于数组的实现

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

int main() {
    int sum = 0;
    int min = 100, max = 0;
    for (int i = 0; i < 7; i++) {
        int score;
        cin >> score;
        if (score > max) max = sc
        if (score < min) min = sc
        sum += score;
    }
    cout << "Score: " << (sum - m
    return 0;
}
```

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

int main()
{
    double avg = 0.0;
    int score[7], min = 100, max = 0;
    for (int i=0; i<7; i++) {
        cin >> score[i];
        if (min > score[i])
            min = score[i];
        if (max < score[i])
            max = score[i];
        avg += score[i];
    }
    avg -= (max + min);
    avg /= 5;
    cout << "avg = " << avg << endl;
    return 0;
}
```

【任务5.1】 基于数组的实现（2）

将数据处理与IO操作分离

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

int main()
{
    int score[7];

    // 1. 先完成所有输入和保存
    for (int i = 0; i < 7; i++)
        cin >> score[i];

    // 2. 然后再进行统计计算
    int min = 100, max = 0;
    double avg = 0.0;
    for (int i = 0; i < 7; i++) {
        if (min > score[i])
            min = score[i];
        if (max < score[i])
            max = score[i];
        avg += score[i];
    }
    avg -= (max + min);
    avg /= 5;

    cout << "avg = " << avg << endl;
    return 0;
}
```

建议尽量采用这种“分离IO”的实现！

【任务5.2】 歌手大赛评分系统“升级版”

某电视台举办歌手大奖赛，邀请了一些评委给歌手打分。在计算歌手的平均得分时，为了公平起见，要把这些评委给出的最高分、最低分都去掉，只将剩下的评委给出的评分进行平均。

为了提高效率和保存记录，在评委报出成绩后，工作人员将分数录入计算机中，通过专门编写的程序来计算和输出歌手的最终得分。

请你编写程序来完成这个任务。设评委给分区间为0~100分。

输入数据为：第一行为整数，表示评委数量N；后续若干行整数，分别表示N个评委给出的评分。

能否让数组大小是一个运行时确定数值的“变量”？

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

int main()
{
    int N; cin >> N;
    int score[N];

    // 1. 先完成所有输入和保存
    for (int i = 0; i < N; i++)
        cin >> score[i];

    // 2. 然后再进行统计计算
```

```cpp
1  #include <iostream>
2  using namespac
3
4  int main()
5  {
6      int N; cin
7      int score[N];
8      for (int i = 0; i < N; i++)
9          cin >> score[i];
```

int N

表达式必须含有常量值 C/C++(28)

EX13.cpp(7, 15): 变量 "N" (已声明 所在行数:6) 的值不可用作常量

查看问题　快速修复... (Ctrl+.)

然而，...

使用c++标准库的可变长“数组”（向量）
vector<T>

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

int main() {
    int N; cin >> N;
    vector<int> score(N);

    // 1. 先完成所有输入和保存
    for (int i = 0; i < N; i++)
        cin >> score[i];

    // 2. 然后再进行统计计算
    int min = 100, max = 0;
    double avg = 0.0;
    for (int i = 0; i < N; i++) {
        if (min > score[i])
            min = score[i];
        if (max < score[i])
            max = score[i];
        avg += score[i];
    }
    avg -= (max + min);
    avg /= (N - 2);

    cout << "avg = " << avg << endl;
    return 0;
}
```

【任务5.3】未知长度序列求和

2314. N长的序列求和

我的提交　所有提交　编辑

1000 ms　256 MB　?

题目描述

有一个长度为n的序列，请你按照顺序，选择一些数，输出他们的和。
一个数满足以下所有要求，才能被选中：
(1) 能被3或者5整除；
(2) 小于1000；
(3) 在这个数出现之前，没有出现过-1。
(4) 当前选中的数字个数小于10个。（即，最多只能选中10个数字）

程序需要持续处理序列中的整数，直到遇到-1才停止。

参考实现源程序

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

int main()
{
    vector<int> list;
    while (1)
    {
        int x;
        cin >> x;
        if (x == -1)
            break;
        list.push_back(x);
    }
    int sum = 0;
    for (int i = 0; i < list.size(); i++)
    {
        int x = list[i];
        if (((x % 3 > 0) && (x % 5 > 0)) ||
            (x >= 1000))
            continue;
        sum += x;
    }
    cout << sum;

    return 0;
}
```

返回list元素数量

将x添加到list的尾部

语法新知识（要点）：

```
while (条件表达式)
{
    // 循环体
}
```

只要条件表达式为真，就执行“循环体”

【任务5.4】输出100以内所有的质数

```
#include <iostream>

using namespace std;


int main() {


    return 0;
}
```

【任务5.4】输出100以内所有的质数

```
#include <iostream> // cout

using namespace std;

int main() {
    for (int i = 2; i <= 100; i++)
        cout << i << ' ';
    cout << endl;
    return 0;
}
```

若满足条件，则

【任务5.4】输出100以内所有的质数

```
#include <iostream> // cout

using namespace std;

int main() {
    for (int i = 2; i <= 100; i++)
        if (IsPrime(i)) cout << i << ' ';
    cout << endl;
    return 0;
}
```

【任务5.4】输出100以内所有的质数

```
#include <iostream> // cout

using namespace std;

bool IsPrime(int n) {
    for (int k = 2; k < n; k++)
        if (n % k == 0) return false;
    return true;
}

int main() {
    for (int i = 2; i <= 100; i++)
        if (IsPrime(i)) cout << i << ' ';
    cout << endl;
    return 0;
}
```

能否再提高计算效率？

【任务5.4】输出100以内所有的质数

```
#include <iostream> // cout
#include <cmath>    // sqrt
using namespace std;

bool IsPrime(int n) {
    for (int k = 2; k <= sqrt(n); k++)
        if (n % k == 0) return false;
    return true;
}

int main() {
    for (int i = 2; i <= 100; i++)
        if (IsPrime(i)) cout << i << ' ';
    cout << endl;
    return 0;
}
```

能否再提高计算效率？

缩小枚举范围
可提高速度

用筛法求100以内的所有质数

“筛法”，也称“埃拉托斯特尼筛法”（Sieve of Eratosthenes）。

想象一下：我们可以将100个数看作沙子和小石头子，其中小石头子代表质数，沙子代表合数（非质数）。这样，如果我们可以找来一个筛子，用它将沙子（合数）筛走，那么，剩下的就是小石头子（质数）了。

注意：根据数学定义，合数（非质数）一定是 2、3、4 …… 的倍数。

筛法的过程（动画演示）

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

筛法求质数的依据

1至100这些自然数可以分为三类：

- 单位数：仅有一个数1。
- 质数：　大于1，且只有1和它自身两个正因数。
- 合数：　除了1和自身以外，还有其他正因数（至少一个）。

除1以外的自然数，只有质数与合数。筛法实际上是筛去合数，留下质数。

因此，执行下面的操作，就可以得到指定范围内的全部质数：

（1）从2开始到100的每一个候选数字放到“筛子”里。

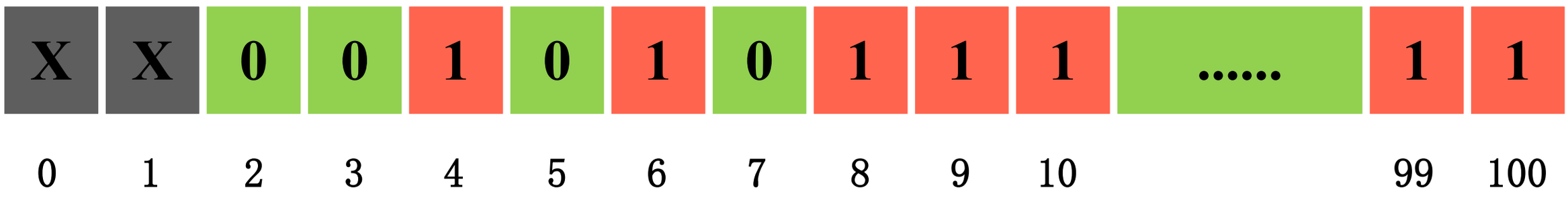
（2）每一轮筛选时都只考虑当前还没有筛掉的数字，并且要从第一个数开始（这个数必定是的质数），把除了该数本身之外的那些倍数全部筛掉！

（3）反复执行上面这个筛选过程，直到把小于SQRT(100)的倍数全部筛掉为止。

筛法求100以内所有质数的实现思路

使用数组来实现筛法：

用数组的下标表示100以内的各个数，数组的元素值表示下标对应的数是否被筛去（即是否是质数）的标志，筛掉的数相应元素值为1。最终剩下的数相应的元素值为0，即为质数。



筛法求质数的源程序

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

int main() {
    int prime[101] = {1, 1};
    for (int d = 2; d <= 10; d++) { // 10 = sqrt(100)
        if (prime[d] == 0) {
            for (int k = d+d; k <= 100; k+=d)
                prime[k] = 1;
        }
    }

    for (int i = 2; i <= 100; i++)
        if (prime[i] == 0) cout << i << ' ';
    cout << endl;

    return 0;
}
```

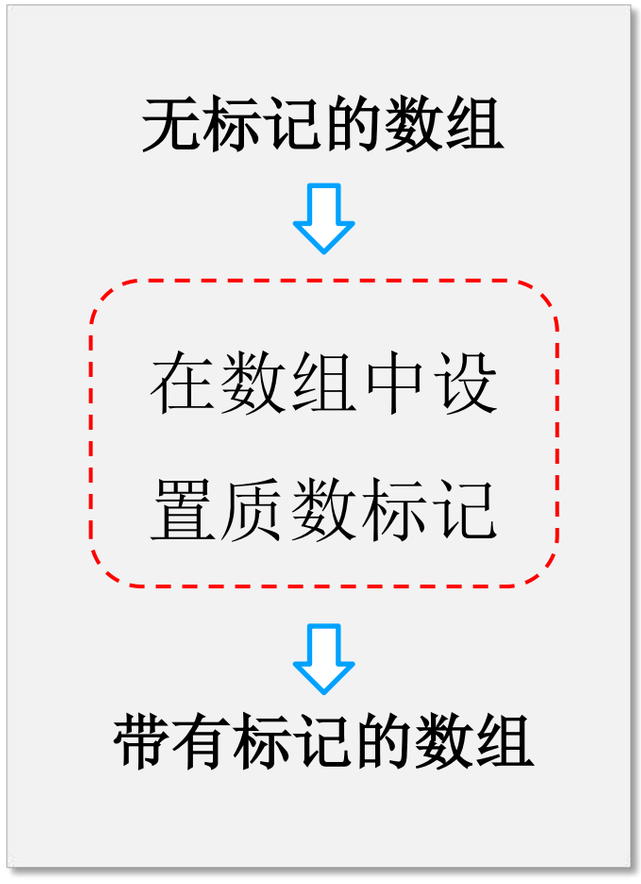
prime[0] = 1, prime[1] = 1,
prime[2] = 0, …

用数组做“记号”（记录）

根据数组中的“记号”决定是否要输出

【代码重构】提炼函数，增强复用性

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

int main() {
    int prime[101] = {1, 1};
    for (int d = 2; d <= 10; d++) { // 10 = sqrt(100)
        if (prime[d] == 0) {
            for (int k = d+d; k <= 100; k+=d)
                prime[k] = 1;
        }
    }

    for (int i = 2; i <= 100; i++)
        if (prime[i] == 0) cout << i << ' ';
    cout << endl;

    return 0;
}
```



【代码重构】提炼函数，增强复用性

无标记的数组

⇩

在数组中设置质数标记

⇩

带有标记的数组

```cpp
#include <iostream> // cin, cout
#include <cmath>    // sqrt
using namespace std;

// void FindPrimes(int array[101], int n)
// void FindPrimes(int array[], int n)
void FindPrimes(int* array, int n) {
    for (int d = 2; d <= sqrt(n); d++) {
        if (array[d] == 0) {
            for (int k = d+d; k <= n; k+=d)
                array[k] = 1;
        }
    }
}
int main() {
    int prime[101] = {1, 1};
    FindPrimes(prime, 100);
    for (int i = 2; i <= 100; i++)
        if (prime[i] == 0) cout << i << ' ';
    cout << endl;

    return 0;
}
```

虽然也可以，但不建议使用。

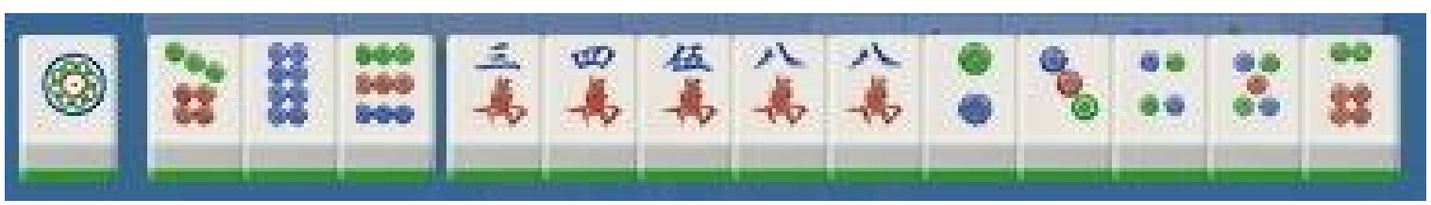
语法知识

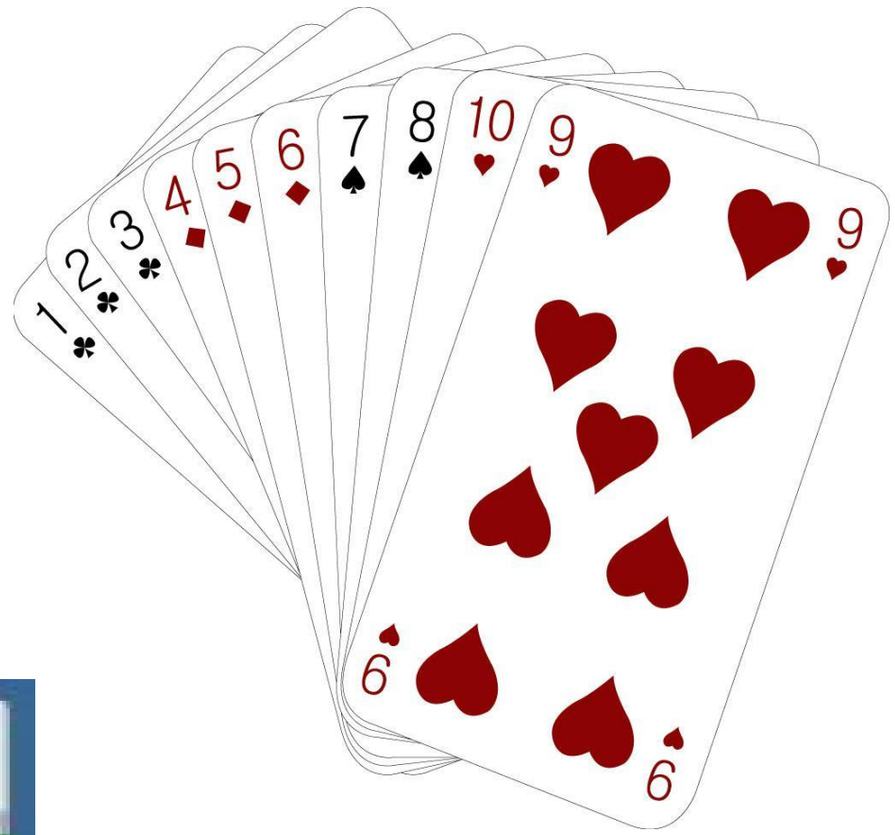
数组作函数形参时的三种写法

【任务5.5】 在数组中查找指定元素

如何在一个数组（一批数据）中查找某个指定的元素，即回答该元素是否存在？如果存在，它在哪里（对应的数组下标是什么）？

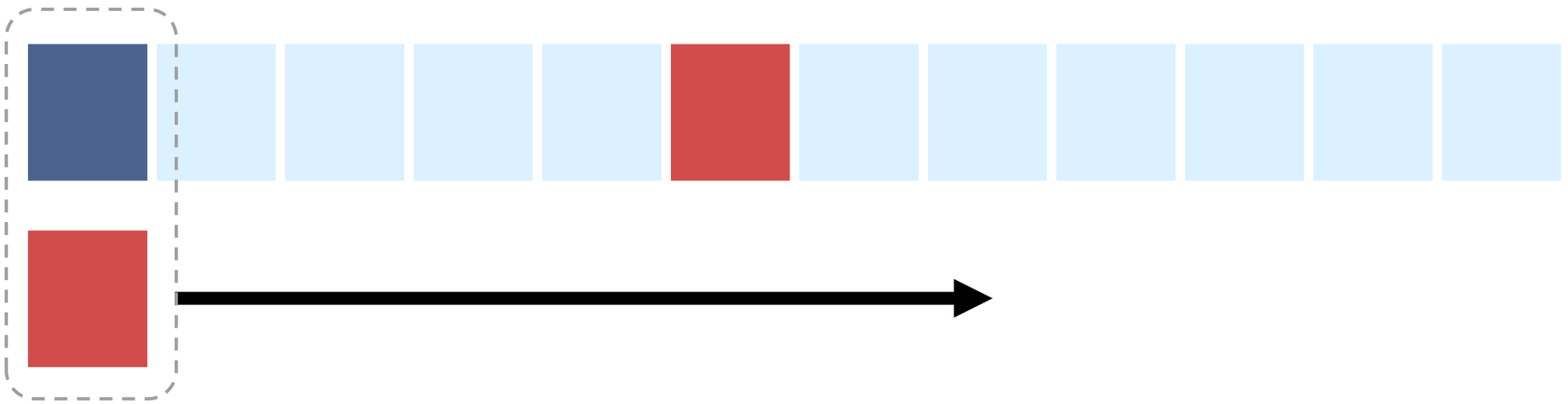
查找某个指定目标，是生活中一种常见的需求

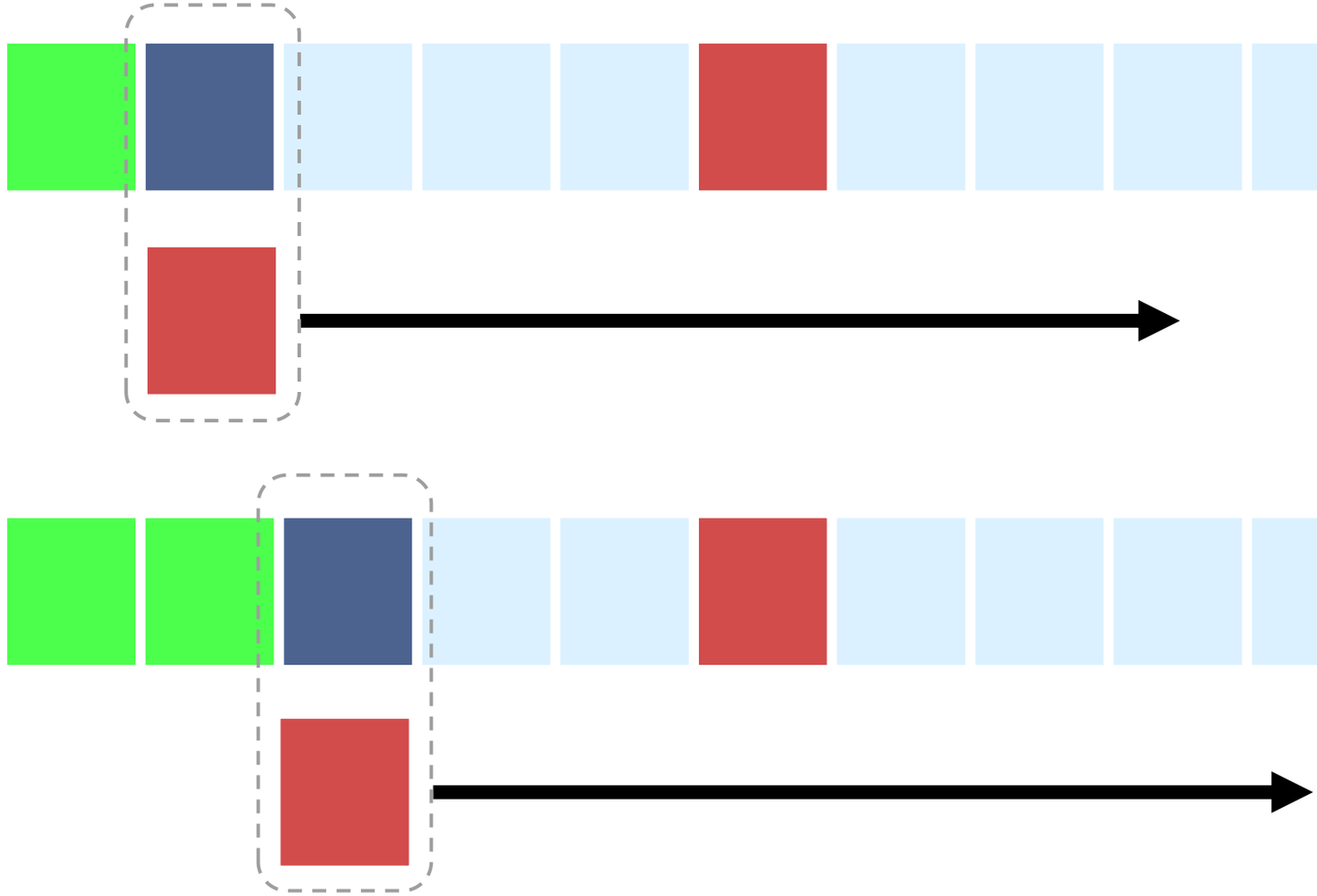




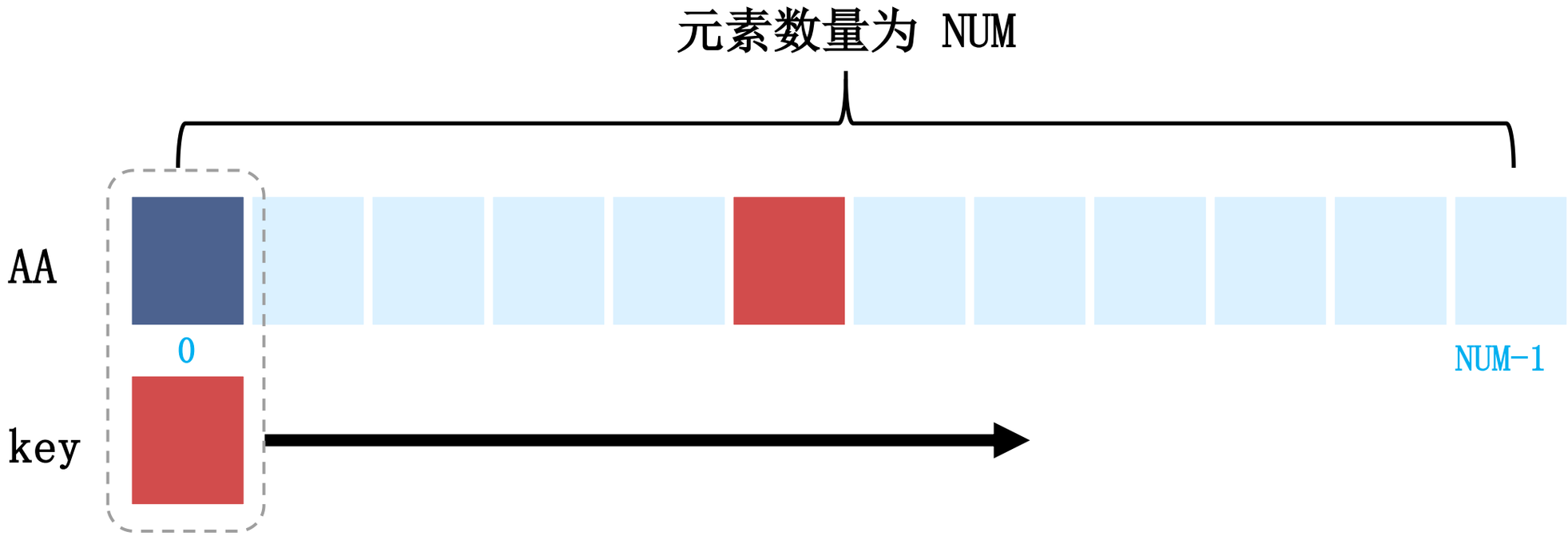
解题思路：线性查找（枚举思想）

依次序进行比较（判断）





算法实现（代码片段）




```
for (int i=0; i<NUM; i++)
    if (AA[i] == key)
        return i;  // found!
```

还有更快些的算法吗

有！答案就在 …….　　请把教材翻到第134页

《楞严经》卷二：

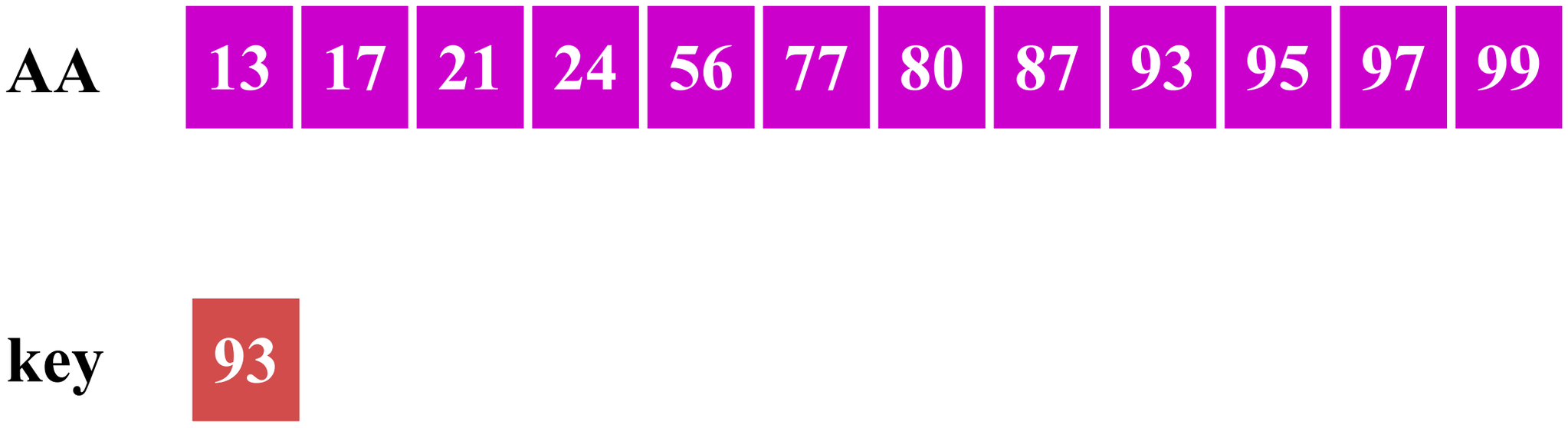
如人以手，指月示人，彼人因指，当应看月。若复观指，以为月体，此人岂唯亡失月轮，亦亡其指。

更快的算法 ——— 折半查找（二分查找）

折半查找

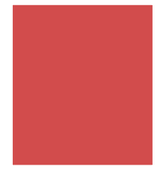
折半查找的前提条件：数组元素是有序存放的。

不妨设 AA[0] < … < AA[NUM-1]

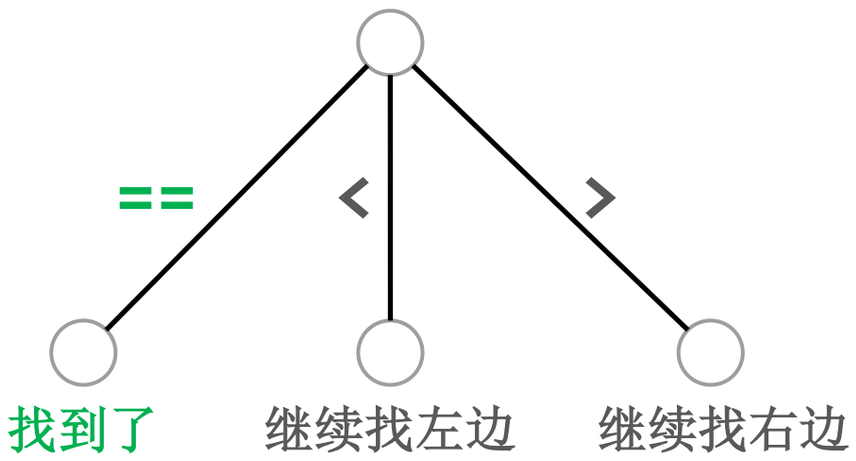


折半查找的思路

key

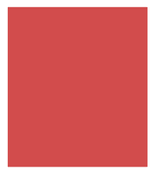


```
key  ?  AA[middle]
```



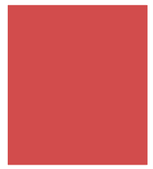
折半查找的思路

key



```
1. if (key == AA[middle]) ...
2. if (key  < AA[middle]) ...
3. if (key  > AA[middle]) ...
```

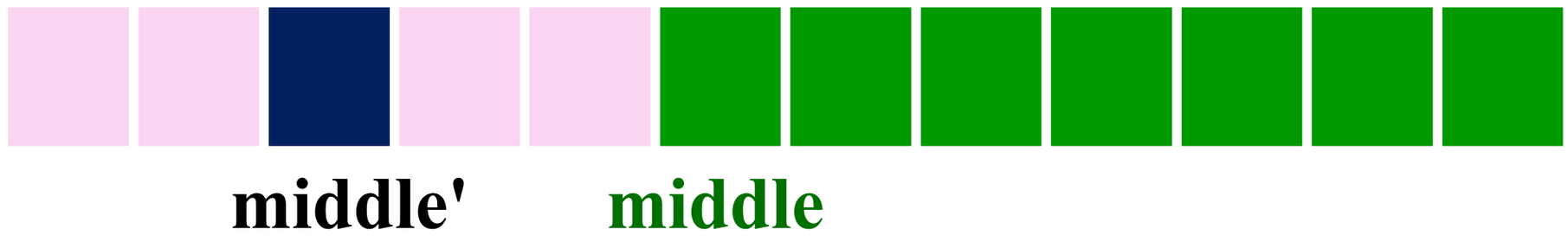
折半查找的思路



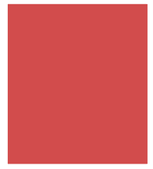

```
1. if (key == AA[middle]) ...
```

找到了！

折半查找的思路



key



```
2. if (key  < AA[middle]) ...
```

KEY不在右半部分
对左半部分重复之前操作

折半查找的思路


```
3. if (key  > AA[middle]) ...
```

KEY不在左半部分
对右半部分重复之前操作

何时停止？

AA

key

（数组）片段为空！“没有剩余的元素”！

```cpp
#include <iostream>  // cin, cout
#include <iomanip>   // setw - 设置显示宽度
using namespace std;

//（此处代码见下一页）

int main() {
    const int NUM = 100; // NUM是整型常量
    int a[NUM]; // 数组大小可以使用整型常量
    for (int i=0; i<NUM; i++) {
        a[i] = i*i+1;
        cout << setw(4) << a[i] << ((i+1) % 10 == 0 ? '\n' : ' ');
    }

    int searchKey;
    cout << "请输入一个待查正整数: ";
    cin >> searchKey;

    int b = 0;
    b = BinarySearch(a, searchKey, 0, NUM-1);

    if (b != -1)
        cout << "查到该数在数组中为: a[" << b << "]\n";
    else
        cout << "数组中无此数!\n";

    return 0;
}
```

这段代码设置了数组各元素的值，方便算法测试。

表示后面数据显示的宽度为4个字符，并且右对齐。

如果找到，则返回相应的下标；如果没有，则返回-1。

```
int BinarySearch(int AA[], int Key,
                 int low, int high) {
    int middle = 0;
    while (low <= high) {
        middle = (low + high) / 2;
        if (Key == AA[middle])
            return middle;
        else if (Key < AA[middle])
            high = middle - 1;
        else
            low = middle + 1;
    }
    return -1; // 数组中没有Key
}
```

```
while (条件表达式)
{
   // 循环体
}
```

只要条件表达式为真，就执行“循环体”

注意：这个实现与之前的分析不同

FOR ↔ WHILE

```
for (expr1; expr2; expr3)
{
    expr4;
}
```

要求 expr2 返回或转换为布尔值

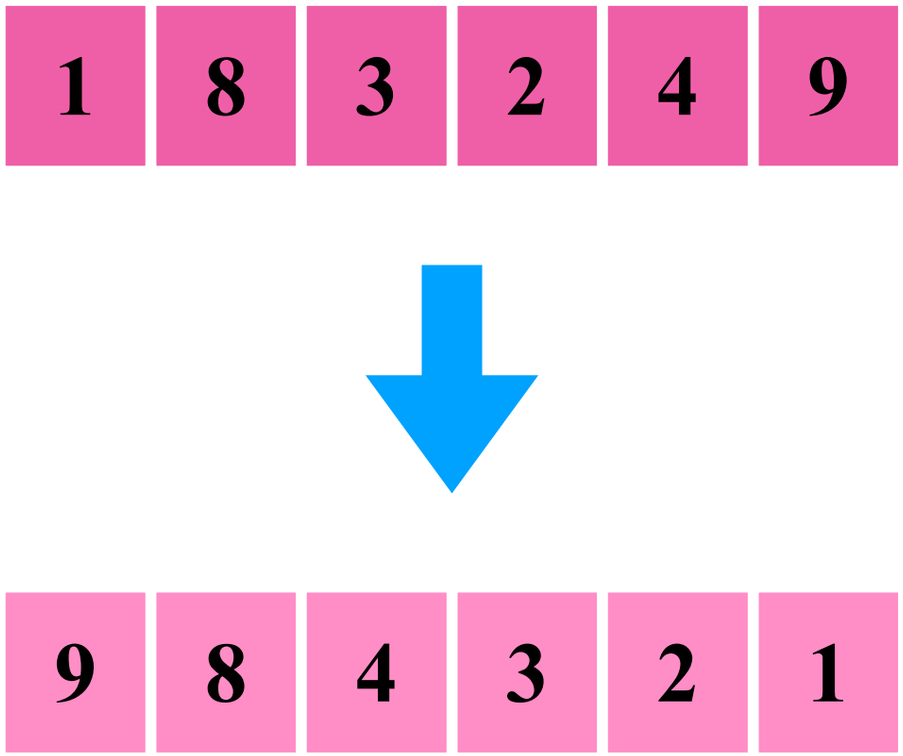
（转换可由编译器自动完成）

```
expr1;
for (; expr2;)
{
    expr4;
    expr3;
}
```

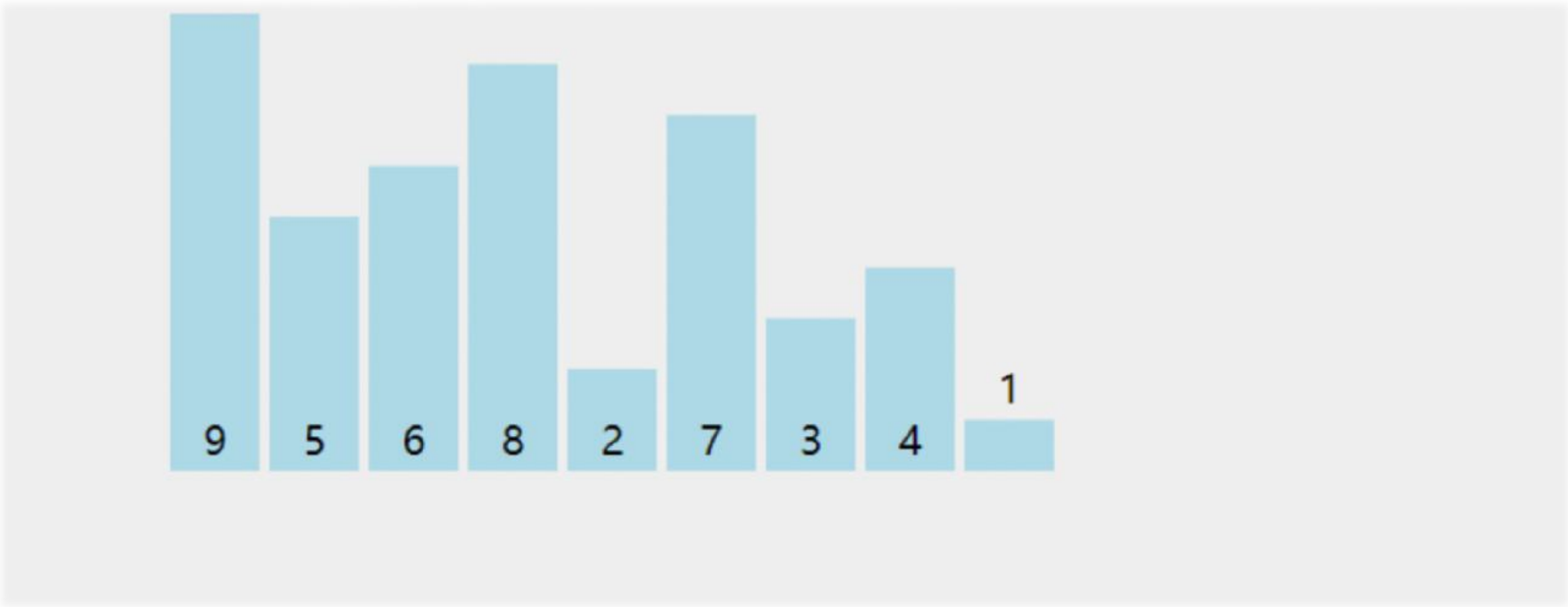
```
expr1;
while (expr2)
{
    expr4;
    expr3;
}
```

【任务5.6】将数组中的元素按大小排序

任务要求：将若干个数从大到小排序，然后输出排序后的结果。



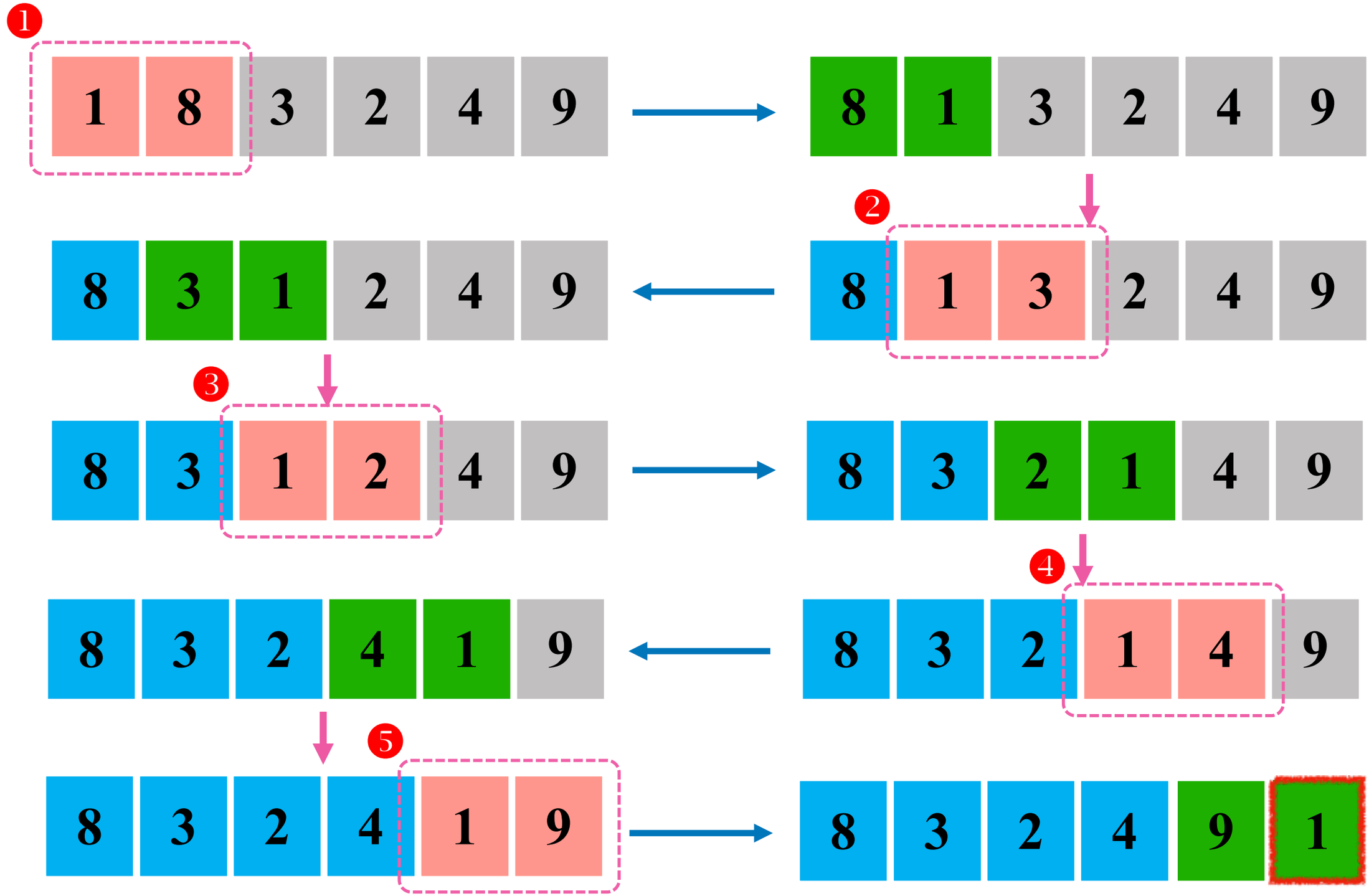
看图作“文”：看动画，写算法



冒泡排序（起泡排序）

冒泡排序的算法思路

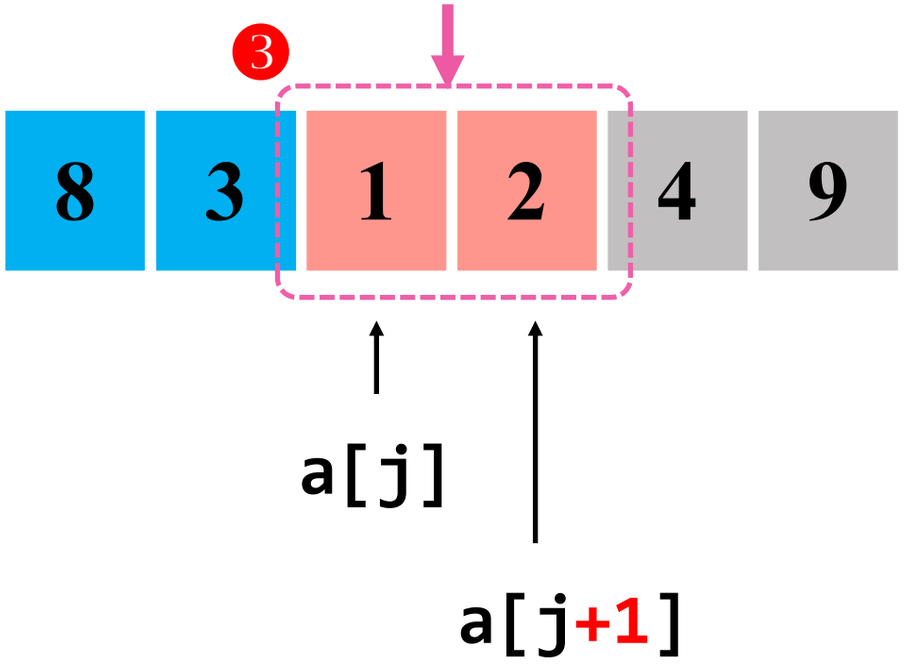
相邻两个元素比较大小，将小数调换到右边

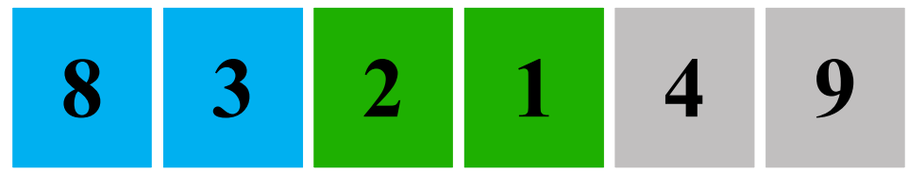


冒泡排序的算法思路

不妨设数组名为 a

总结规律，解决“循环通项”





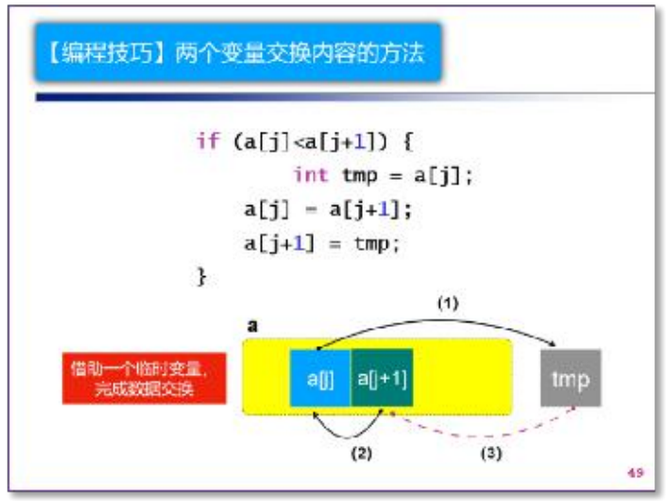
交换了 a[j]，a[j+1]的“位置”？“数值”？

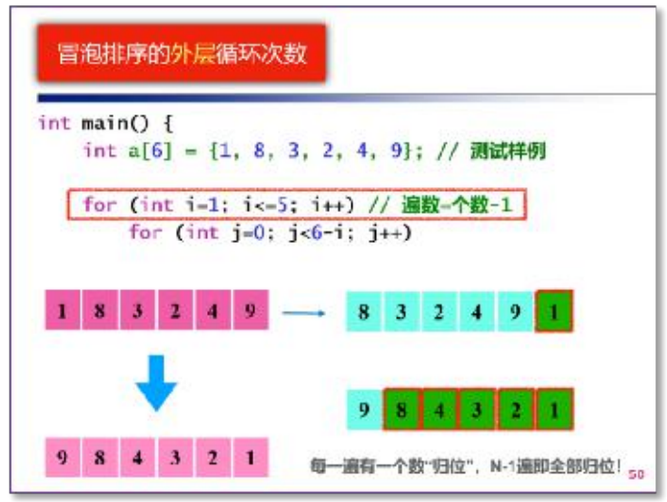
借助辅助变量，完成变量内容交换

```
int tmp = a[j];
a[j]    = a[j+1];
a[j+1]  = tmp;
```

```cpp
#include <iostream> // cout
using namespace std;

int main() {
    int a[6] = {1, 8, 3, 2, 4, 9}; // 测试样例

    for (int i=1; i<=5; i++) // 遍数=个数-1
        for (int j=0; j<6-i; j++) {
            if (a[j]<a[j+1]) {
                int tmp = a[j];
                a[j]    = a[j+1];
                a[j+1]  = tmp;
            }
        }

    for (int i=0; i<6; i++)
        cout << a[i] << ' ';
    cout << endl;

    return 0;
}
```

要点详解

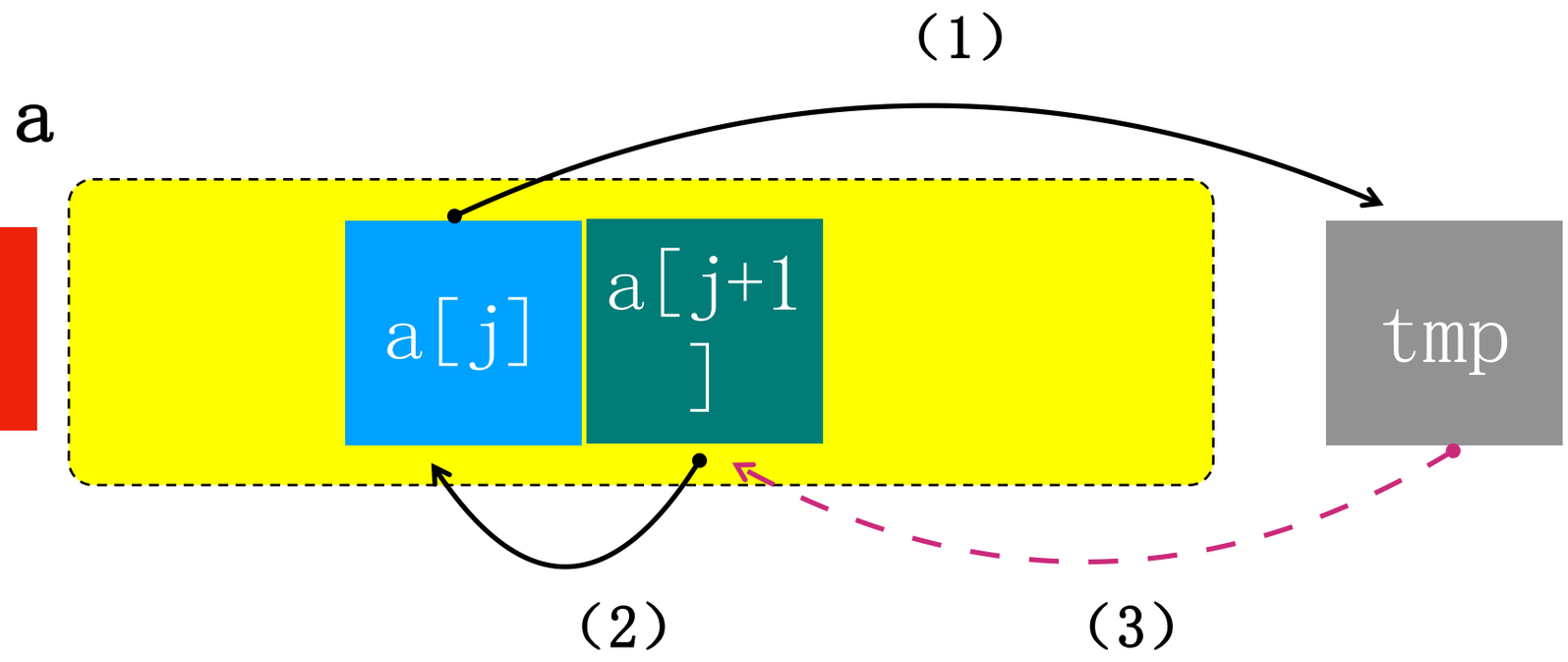




【编程技巧】两个变量交换内容的方法

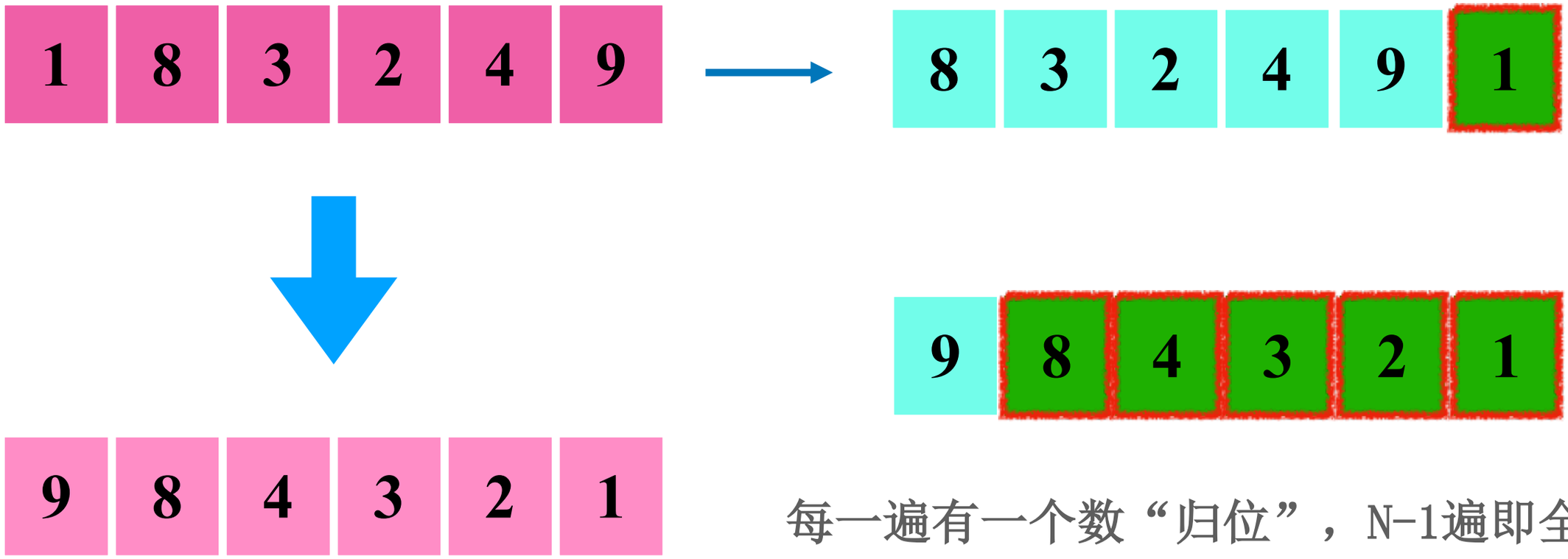
```
if (a[j]<a[j+1]) {
    int tmp = a[j];
    a[j]    = a[j+1];
    a[j+1]  = tmp;
}
```

借助一个临时变量，
完成数据交换



冒泡排序的外层循环次数

```
int main() {
    int a[6] = {1, 8, 3, 2, 4, 9}; // 测试样例

    for (int i=1; i<=5; i++) // 遍数=个数-1
        for (int j=0; j<6-i; j++)
```



每一遍有一个数“归位”，N-1遍即全部归位！

冒泡排序的内层循环次数

```
int main() {
    int a[6] = {1, 8, 3, 2, 4, 9}; // 测试样例

    for (int i=1; i<=5; i++) // 遍数=个数-1
        for (int j=0; j<6-i; j++)
```

写循环的通项表达式时，可以用特例来推演：

第一遍时，i = 1，内层循环（相邻元素两两比较）是从下标 0 处元素开始的，一共进行了 5 次比较，所以内层循环的范围为 0 ~ 4，即 [0 .. 6-i)。

【任务5.7】合伙捕鱼

A、B、C、D、E 五人合伙夜间捕鱼，凌晨时都疲惫不堪，各自在湖边的树丛中找地方睡着了。日上三竿，A第一个醒来，他将鱼平分作五份，把多余的一条扔回湖中，拿自己的一份回家去了。B第二个醒来，也将鱼平分为五份，扔掉多余的一条，只拿走自己的一份。接着　C、D、E 依次醒来，也都按同样的办法分鱼。

问：这五个人至少合伙捕到多少条鱼？每个人醒来后，看到的鱼数分别是多少条？

【任务5.7】合伙捕鱼

为了解决这种类型的问题，需要运用“递推”思想，要掌握用“数组”来实现递推过程的编程技巧。

接下来，我们先介绍“递推”的算法思想，再通过两个小例子来说明递推思想的运用和编程实现，最后再回到本任务，解决捕鱼问题。

递推：先分析，后归纳，求通项

递推是计算机数值计算中的一个重要算法，可以将复杂的运算化为若干重复的简单运算，以充分发挥计算机长于重复处理的特点。

通常，使用循环结构来实现重复处理。

解决此类问题的关键是：分析简单情况，归纳总结出前后项的关系（通项公式）。

递推示例：求自然数的阶乘

任务分析：

令fact(n)表示n的阶乘，依据后项与前项的关系，可以写出下面的“递推”公式：

```
fact(1) = 1                  --- 起始条件（边界条件）
fact(n) = fact(n-1) * n      --- 通项公式
```

算法实现：

根据前后项的关系，从已知推导未知。显然，用循环结构来实现这种递推关系是非常自然的。

递推示例：求自然数的阶乘

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

int main() {
    int N;
    cout << "please input N (N<10):";
    cin >> N;
    int fact[10] = {0, 1}; // 1! = 1 是递推的起点
    for (int n=2; n<=N; n++)
        fact[n] = fact[n-1] * n;  // 阶乘的递推公式
    cout << "fact(" << N << ") = " << fact[N] << endl;
    return 0;
}
```

【任务5.7】合伙捕鱼的解题思路

假定A、B、C、D、E 五人的编号分别为1、2、3、4、5，整数数组fish[k] 表示第 k 个人所看到的鱼数。fish[1] 表示A所看到的鱼数，fish[2] 表示 B 所看到的鱼数……

```
fish[1]  A所看到的鱼数，合伙捕到鱼的总数
fish[2] = (fish[1] - 1) / 5 * 4   B所看到的鱼数
fish[3] = (fish[2] - 1) / 5 * 4   C所看到的鱼数
fish[4] = (fish[3] - 1) / 5 * 4   D所看到的鱼数
fish[5] = (fish[4] - 1) / 5 * 4   E所看到的鱼数
```

【任务5.7】合伙捕鱼的解题思路

写成一般式

$$fish[i] = (fish[i-1] - 1) / 5 * 4$$

$$i = 2, 3, \cdots, 5$$

这个公式可用于已知 A 看到的鱼数去推算 B 看到的，再推算 C 看到的，…….。

现在要求的是 A 看到的。需要倒过来：即先得到E看到的，再反推D看到的，……，直到A看到的。为此将上式改写为：

$$fish[i-1] = fish[i] / 4 * 5 + 1$$

$$i = 5, 4, \cdots, 2$$

算法的通项分析

1．当 i = 5 时，fish[i] 表示 E 醒来所看到的鱼数，该数应满足被 5 整除后余 1，所以初值设为 1 + 5

2. 当 i = 5 时，fish[i-1] 表示 D 醒来所看到的鱼数，这个数要满足

```
fish[ 4 ] = fish[ 5 ]  / 4 * 5 + 1
```

显然，fish[4] 必须是整数，所以fish[5]必须 满足

```
fish[ 5 ] % 4 == 0
```

这个结论同样可以用至 fish[3], fish[2] 和 fish[1]

算法的通项分析

3．按题意要求 5 人合伙捕到的最少鱼数，可以从小往大枚举。

即：可以先让 E 所看到的鱼数最少为 6 条，即 fish[5] 初始化为 6 来试，之后每次增加 5 再试，直至递推到 fish[1]，均为整数。

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

int main() {
    // 记录每人醒来后看到的鱼数
    int i = 0, fish[6] = {1, 1, 1, 1, 1, 1};
    do {
        fish[5] = fish[5] + 5; // 让E看到的鱼数增5
        for (i=4; i>=1; i--) {
            if (fish[i+1] % 4 != 0)
                break;
            else
                fish[i] = fish[i+1] / 4 * 5 + 1;
        }
    } while(i >= 1); // 当 i>=1 继续做do循环

    for (i=1; i<=5; i++)
        cout << fish[i] << endl;
    return 0;
}
```

语法新知识（要点）

:

```
do{
    // 循环体
}while(条件表达式)
```

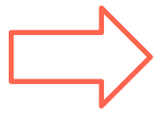
条件表达式为真时，就持续执行“循环体”

输出结果

```
3121
2496
1996
1596
1276
```

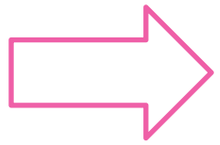
FOR ⟷ WHILE ⟷ DO-WHILE

```
for (expr1; expr2; expr3)
{
    expr4;
}
```



```
expr1;
do {
    if (expr2) { expr4; expr3; }
    else break;
} while (expr2);
```

```
do {
    statements;
} while (cond);
```



```
statements;
while (cond) statements;
```

```
for (;;) {
    statements;
    if (!cond) break;
}
```

以下两种写法效果相同：

```
while (cond) statements;
for (;cond;) statements;
```

【编程技巧】第2种实现方式

课后阅读

```
int main()  {
    int fisher[6];  // 五个人看到的鱼数
    for (int num = 6;  ; num += 5) {  // 对各种可能性进行枚举
        int n;
        for (n=5; n>=1; n--)  { // 对五个人看到的鱼数进行递推
            if (n == 5)
                fisher[n] = num;
            else  {   /// fisher[n+1] = (fisher[n] - 1) / 5 * 4; →
                if (fisher[n+1] % 4 != 0)
                    break; // 鱼数num不满足条件，要尝试下一个num，停止递推
                else  /// 下式所得fisher[n]必然满足 fisher[n] % 5 == 1
                    fisher[n] = fisher[n+1] / 4 * 5 + 1;
            }
        }
        if (n == 0) // 说明鱼数num满足所有人的条件
            break;  // 找到最小鱼数了，停止枚举
    }
    // 输出各个人看到的鱼数
    for (int n=1; n<=5; n++)
        cout << "fisher[" << n << "] = " << fisher[n] << endl;
    return 0;
}
```

【编程技巧】第3种实现方式

课后阅读

```
int main()   {
    int fisher[6];  // 五个人看到的鱼数
    for (int num = 6; ; num += 5) {  // 对各种可能性进行枚举
        int n;
        for (n=1; n<=5; n++) {  // 对五个人看到的鱼数进行递推
            if (n == 1)
                fisher[n] = num;
            else  /// 下式所得fisher[n] 必然是整数
                fisher[n] = (fisher[n-1] - 1) / 5 * 4;

            if (fisher[n] % 5 != 1)
                break; // 鱼数num不满足条件，要尝试下一个num，停止递推

        }
        if (n == 6) // 说明鱼数num满足所有人的条件
            break;  // 找到最小鱼数了，停止枚举
    }
    // 输出各个人看到的鱼数
    for (int n=1; n<=5; n++)
        cout << "fisher[" << n << "] = " << fisher[n] << endl;
    return 0;
}
```

【编程技巧】第4种实现方式

课后阅读

```
int main()   {
    int fisher[6];  // 五个人看到的鱼数
        for (int num = 6; ; num += 5) // 对各种可能性进行枚举
        if (IsOK(num, fisher))
            break; // found
```

```
    // 输出各个人看到的鱼数
    for (int n=1; n<=5; n++)
        cout << "fisher[" << n << "] = " << fisher[n] << endl;
    return 0;
}
```

【编程技巧】第4种实现方式

课后阅读

```
bool IsOK(int num, int fisher[6]) {
    for (int n=1; n<=5; n++) {
        // 对五个人看到的鱼数进行递推
        if (n == 1)
            fisher[n] = num;
        else  // 下式所得fisher[n]必然是整数
            fisher[n] = (fisher[n-1] - 1) / 5 * 4;
        if (fisher[n] % 5 != 1)
            return false; // 鱼数num不满足条件，停止递推
    }
    return true;
}
```

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

int main()
{
    char msg[] = "END. See you later!";
    cout << msg << endl;
    return 0;
}
```