[5] 数组与数据批量存储

深入浅出程序设计竞赛
第 1 部分 – 语言入门
V 2021-02

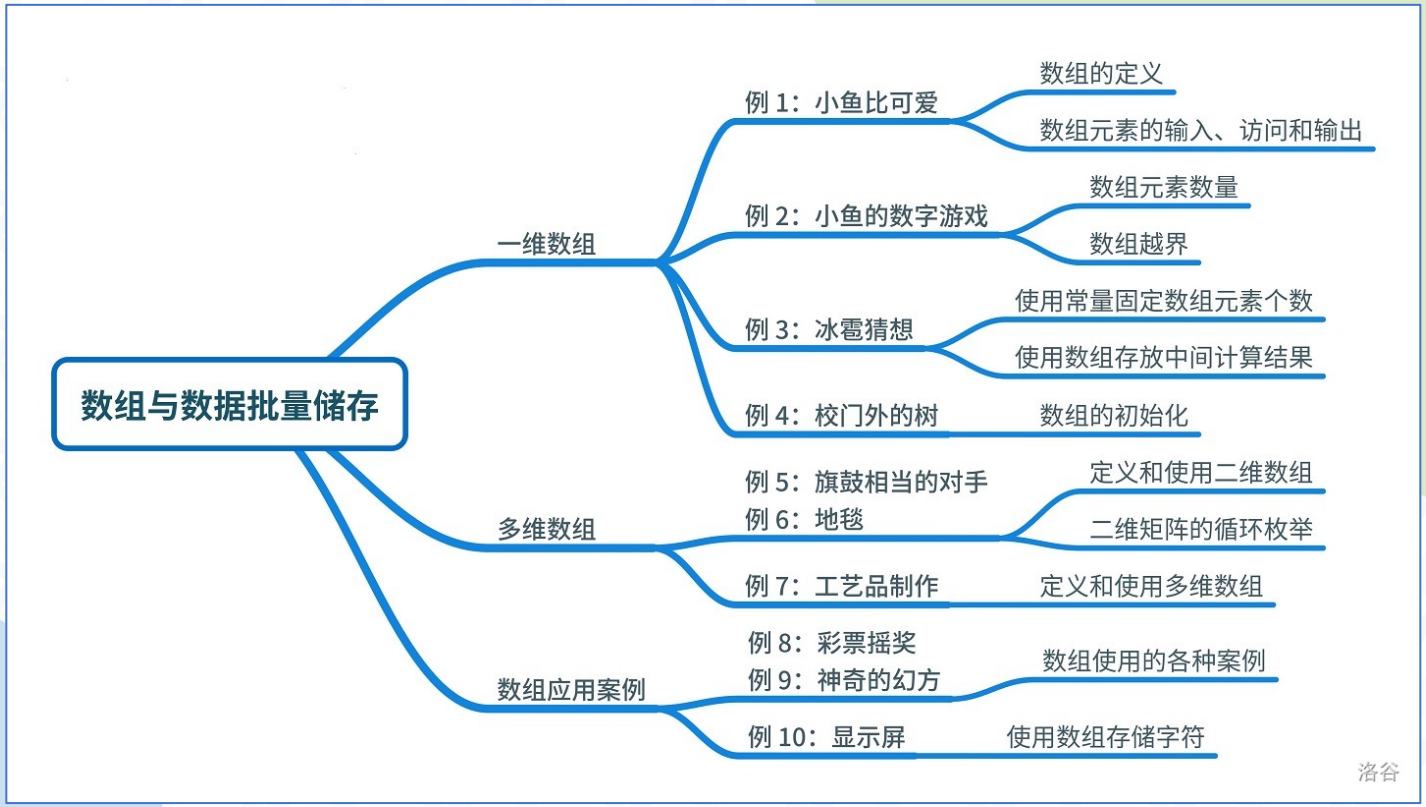
www.luogu.com.cn

洛谷

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图


数组与数据批量储存
一维数组
例 1：小鱼比可爱
数组的定义
数组元素的输入、访问和输出
例 2：小鱼的数字游戏
数组元素数量
数组越界
例 3：冰雹猜想
使用常量固定数组元素个数
使用数组存放中间计算结果
例 4：校门外的树
数组的初始化
多维数组
例 5：旗鼓相当的对手
例 6：地毯
定义和使用二维数组
二维矩阵的循环枚举
例 7：工艺品制作
定义和使用多维数组
数组应用案例
例 8：彩票摇奖
例 9：神奇的幻方
数组使用的各种案例
例 10：显示屏
使用数组存储字符
洛谷

第 5 章 数组与数据批量存储

一维数组

显然，如果要存大量的数据，是很难定义大量的变量。我们可以使用数组来存储和使用批量的数据

请翻至课本 P67

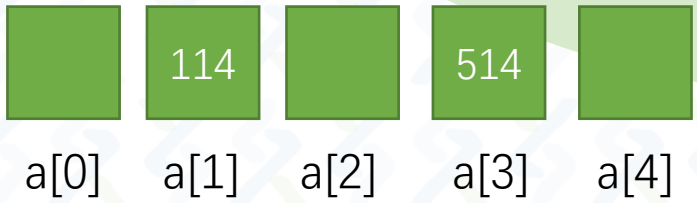
一维数组

数组 可用于存储大量数据，相当于一次性定义多个变量。

定义时方括号里的数字为数组大小，使用时方括号里的数字为下标。

例如，定义名字为 a 类型为 int。数量是 5 的数组：

```
// 数据型数组变量名称[元素个数];
// 例如
int a[5]; // a[0],a[1],a[2],a[3],a[4]
//数组可以赋值，也可以访问
a[1] = 114; a[3] = 514; int b = a[3];
```



数组元素查找

例子

给出 n 个（不超过 100 个）正整数，请输出 x 第一次出现的位置。

```
5
1 2 3 2 1
2
```

```
2
```

数组元素查找

定义 a 数组用于存储数据。注意定义数组的数量时要超过 100。

a[0] 是第 0 个元素（注意数组是从 0 开始计数），a[1] 是第 1 个元素，a[i] 是第 i 个元素。

使用 for 循环遍历数组，如当前数据（第 i 个数字也就是 a[i]）与 x 相同则直接输出当前下标 i。注意输出后使用 break 跳出循环。

```
int n, a[105], x;
cin >> n;
for(int i = 1; i <= n; i++)
    cin >> a[i];
cin >> x;
for(int i = 1; i <= n; i++)
    if(a[i] == x) {
        cout << i; break;
    }
```

小鱼比可爱

例 5.1（洛谷 P1428）

已知有 n 只鱼，不超过 100 只，从左到右排成一排，每只鱼都有它的可爱程度（整数）

所有的鱼头都朝向左边，只能看见在它左边的鱼的可爱程度

每只鱼都想知道它左边右多少小鱼可爱程度小于它？

```
6
4 3 0 5 1 2
```

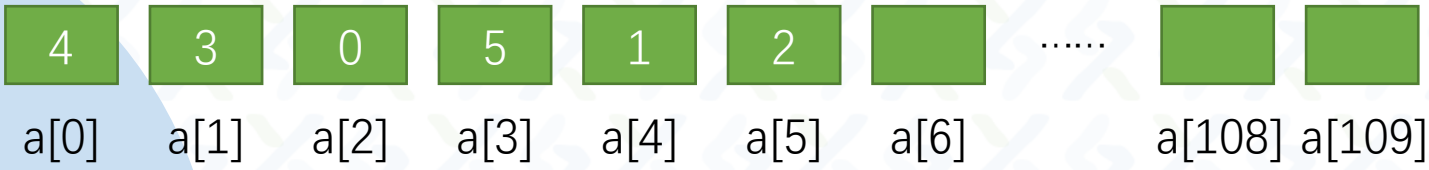
```
0 0 0 3 1 2
```

小鱼比可爱

定义数组 a 存储小鱼的可爱度，使用变量 tot 统计有多少只小鱼不如当前小鱼可爱。for 循环 i 从 0 到 n-1 读入 a[i]。

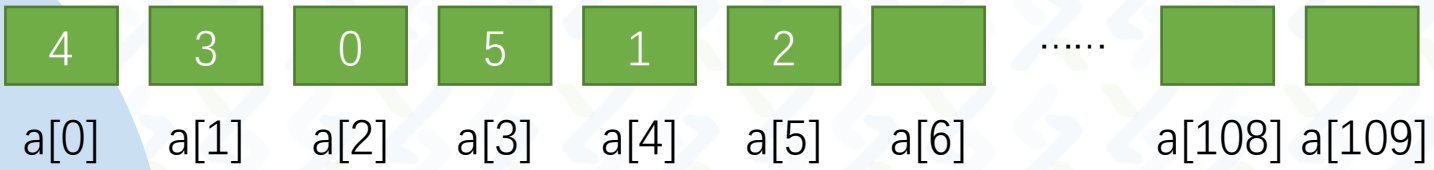
- 第一重 for 循环表示对当前小鱼（第 i 只）进行统计答案，因此变量 i 方向为 0 到 n-1；
- 第二重 for 循环用于统计每一只小鱼（a[j]）不如当前小鱼（a[i]）可爱，所以方向从 i-1 到 0；

注意数组 a[x] 的下标从 0 开始到 x-1 结束，所以实际定义数组时应稍微定义大一点。题面提到不超过 100 只小鱼，所以数组可以定义到 a[110]。



小鱼比可爱

```cpp
#include<iostream>
using namespace std;
int main() {
    int a[110], n;
    cin >> n;
    for (int i = 0; i < n; i++) // 读入每条鱼的可爱值
        cin >> a[i];
    for (int i = 0; i < n; i++) { // 枚举n条鱼
        int cnt = 0;
        for (int j = i - 1; j >= 0; j--)  // 从第i个位置倒着往前找
            if (a[j] < a[i])
                cnt++; // 如果找到比第i条鱼没有比不上，就增加计数器
        cout << cnt << ' ';
    }
    return 0;
}
```



小鱼的数字游戏

例 5.2（洛谷 P1427）

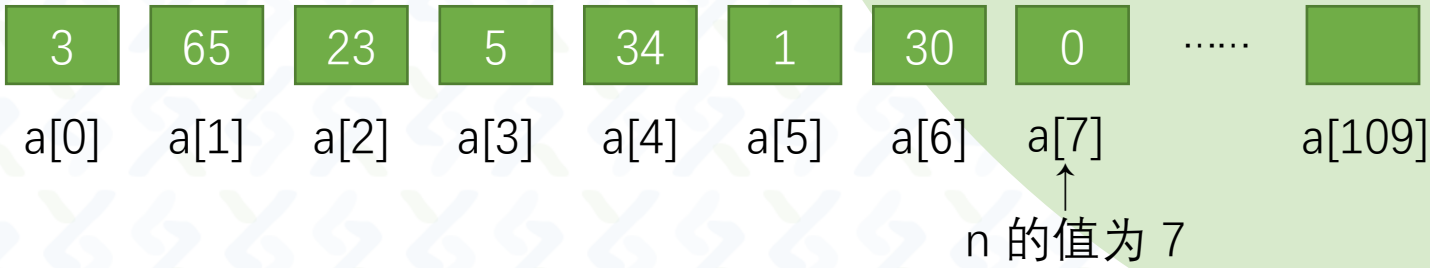
小鱼最近被要求参加一个数字游戏，要求它把看到的一串数字ai（长度不一定，不超过 100，以 0 结束），记住了然后反着念出来（不包括 0）。请帮小鱼解决这个问题。

```
3 65 23 5 34 1 30 0
```

```
30 1 34 5 23 65 3
```

小鱼的数字游戏

和前一题类似，读入数字，从 a[0] 存到 a[n-1]，然后从 a[n-1]开始逆序输出。使用以前学习过的 do-while 循环。




```
int n = 0, tmp, a[110];
do {
    cin >> tmp;
    a[n] = tmp;
    n++;
} while (tmp != 0); //用于使循环中止
n--;//因为下标最大为n-1
while (n--)
    cout << a[n] << ' ';
```

冰雹猜想

例 5.3（洛谷 P5727）

给出一个正整数n(n≤100)，如果这个数字是奇数，那么将其乘 3 再加 1，否则除以 2。经过若干次循环后，最终都会回到 1。

例如当 n 是 20，变化的过程是 [20, 10, 5, 16, 8, 4, 2, 1]。

根据给定的数字，验证这个猜想，并从最后的 1 开始，倒序输出整个变化序列。

冰雹猜想

提示：按照题目的要求计算数字并依次把答案存到数组中，最后用 for 循环倒序输出。定义数组时数组大小必须为常数，我们使用宏定义 MaxN 为205，这样方便后面修改。

```
#include<iostream>
using namespace std;
#define MAXN 205
int main() {
    int n, num = 0, a[MAXN];
    cin >> n;
    while (n != 1) {
        a[num] = n; num++; // 本行可以替代成a[num++]=n;
        if (n % 2 == 0)n /= 2;
        else n = 3 * n + 1;
    }
    a[num] = 1; // 将最后的1加入到数组中
    for (int i = num; i >= 0; i--) // 倒序输出
        cout << a[i] << ' ';
    return 0;
}
```

数组的初始化

数组如果不初始化就可能存有其他数值，因此我们在使用前必须初始化，可以使用 for 循环依次赋初值，也可以在定义时直接初始化。

我们还可以使用 memset 函数来实现（需要 cstring 头文件）

```
int a[10010]={0}; // 这表示 a 数组中的所有变量初始化为 0。
memset(数组名称,0,sizeof(数组名称));
```

但如果数组过大，这会消耗很多时间。

校门外的树

例 5.4（洛谷 P1047）

某校大门外长度为 L（不超过 10000）的马路上有一排树，每两棵相邻的树之间的间隔都是 1 米。

现给出若干区域，已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。

你的任务是计算将这些树都移走后，马路上还有多少棵树。

```
500 3
150 300
100 200
470 471
```

```
298
```

校门外的树

定义 a 数组表示该位置上是否被砍（0 表示没被砍，1 表示被砍了）

这题需要对 a 数组初始化为 0，因为开始时每个位置都没有被砍。

我们对每一个区域进行处理，把区域内的树砍掉，也就是把 0 变成 1。最终统计数组中 0 的个数。

```
#include <iostream>
// #include <cstring>
using namespace std;
int main() {
    int l, m, tree[10010]={0},a,b,s=0;
    cin >> l >> m;
    // memset(tree, 0, sizeof(tree));
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        for (int j = a; j <= b; j++)
            tree[j] = 1;
    }
    for (int i = 0; i <= l; i++)
        if (tree[i] == 0) s++;
    cout << s << endl;
    return 0;
}
// 下图从 2 砍到 5
// 想一想，为什么 a[5] 没变
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
0	0	1	1	1	0	0

数组元素添加和删除

例子

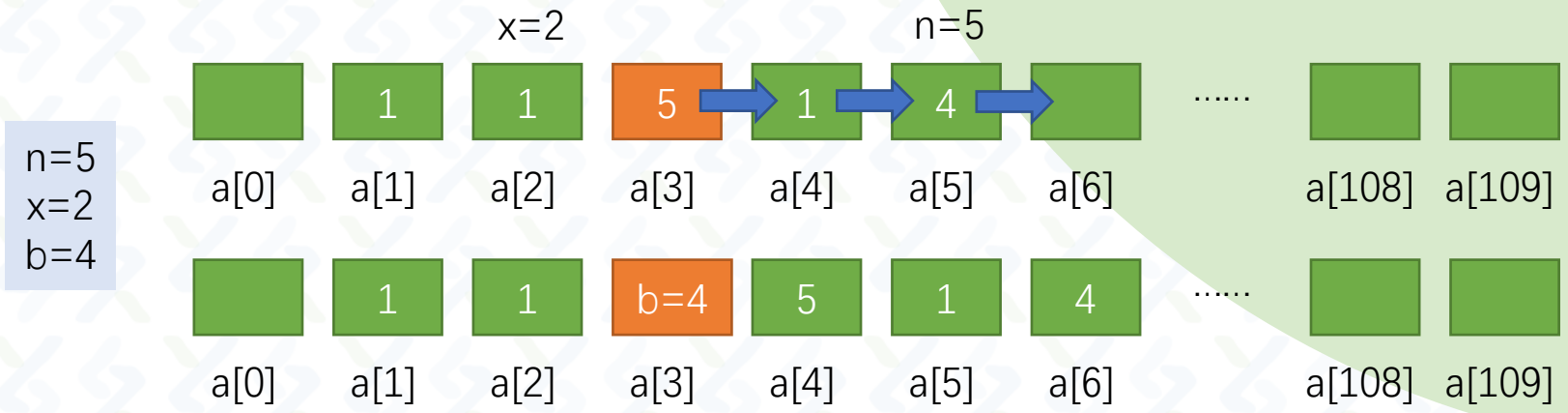
给出 n 个元素的数列，请在第 x 个元素和 x+1 个元素之间插入 b 并输出该数列。

```
5
1 1 5 1 4
2 4
```

```
1 1 4 5 1 4
```

数组元素添加和删除

先把编号为 x+1 到 n 的所有元素向后移动一位，然后在原 x+1 的位置插入 b。注意移动时为了防止元素被覆盖应从后往前处理。




```cpp
#include<iostream>
using namespace std;
int main() {
    int n, a[105], x, b;
    cin >> n;
    for(int i = 1; i <= n; i++)
        cin >> a[i];
    cin >> x >> b;
    for(int i = n; i >= x + 1; i--)
        a[i + 1] = a[i];
    a[x + 1] = b;
    for(int i = 1; i <= n + 1; i++)
        cout << a[i] << ' ';
    return 0;
}
```

校门外的树

判断练习

某校大门外长度为 L（不超过 10000）的马路上有一排间隔 1 米的树。

给出若干可能重叠区域，任一区域的起始和终止点的坐标都是整数。把这些区域中的树（包括端点）移走。最后马路上还有多少棵树。

请问这个程序有几处错误？

```
#include <iostream>
using namespace std;
int main() {
    int l, m, tree[10000], a,b,s=0;
    cin >> l >> m;
    for(int i = 0; i < m; i++) {
        cin >> a >> b;
        for(int j = a; j <= b; j++)
            tree[j] = 1;
    }
    for(int i = 0; i <= l; i++)
        if(tree[i] == 0) s++;
    cout << s << endl;
}
```

校门外的树

在应用数组时最容易犯的错误：

- 在函数内开数组，但是没有初始化（或读入）而直接使用。
- 数组数量没有开够。

```
#include <iostream>
// #include <cstring>
using namespace std;
int main() {
    int l, m,tree[10010]={0},a,b,s=0;
    cin >> l >> m;
    // memset(tree, 0, sizeof(tree));
    for(int i = 0; i < m; i++) {
        cin >> a >> b;
        for(int j = a; j <= b; j++)
            tree[j] = 1;
    }
    for(int i = 0; i <= l; i++)
        if(tree[i] == 0) s++;
    cout << s << endl;
    return 0;
}
```

多维数组

数组也能嵌套数组。如果你需要存储一个表格，那么需要使用多维数组。

请翻至课本 P53

多维数组

如果说一维数组是一排变量，二维数组就是矩阵。

这里以最简单的二维数组为例，多维数组与此类似

```
// 定义方式为 数组类型 数组变量名称[行数][列数];
int a[4][5];
// 可得到一个四行五列的数组（注意下标从 0 开始
```

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

元素查找

例子

给出一个 n 行 m 列的矩阵（n 和 m 均不超过 100），每个矩阵元素是不超过 100 的数字。

然后请问第 x 行第 y 列（从 1 开始编号）的元素是什么？

例如，下面的样例，第 2 行第 3 列的元素是 0。

```
3 4
3 5 3 9
1 3 0 4
2 4 6 8
2 3
```

```
0
```

元素查找

定义这个二维矩阵 a[105][105]。a[0][0] 废弃，最大只能 a[104][104]，输入时下标从 1 开始。

直接使用下标访问 如访问第二行第三列时使用 a[2][3]。

	第0列 a[?][0]	第1列 a[?][1]	第2列 a[?][2]	第3列 a[?][3]	第4列 a[?][4]
第0行 a[0][?]					
第1行 a[1][?]		3	5	3	9
第2行 a[2][?]		1	3	0	4
第3行 a[3][?]		2	4	6	8

```
#include<iostream>
using namespace std;
int main() {
  int a[105][105] = {0};
  int n, m, x, y;
  cin >> n >> m;
  for (int i = 1; i <= n; i++)
    for (int j = 1; j <= m; j++)
      cin >> a[i][j];
  cin >> x >> y;
  cout << a[x][y];
}
```

旗鼓相当的对手

例 5.5（洛谷 P5728）

N 名同学参加语数英三门考试，每门得分不超过 150

若两名学生单科分差不大于 5 分且总分分差不大于 10 分，则称为一对“旗鼓相当的对手”

问 N 名同学中有几对旗鼓相当（同一人可能会跟多名同学结对）

```
3
90 90 90
85 95 90
80 100 91
```

```
2
```

旗鼓相当的对手

定义一个二维数组 a[x][4]。a[x][0],a[x][1],a[x][2],a[x][3] 分别表示第 x 名同学的语数英三门课的成绩和总成绩。

两重循环枚举所有学生对，判断是否满足条件后统计答案。使用 abs 函数判断差距。

	第0列 a[?][0]	第1列 a[?][1]	第2列 a[?][2]	第3列 a[?][3]
第0行 a[0][?]				
第1行 a[1][?]	90	90	90	270
第2行 a[2][?]	85	95	90	270
第3行 a[3][?]	90	100	91	281

```cpp
// 需要 cmath 头文件
int n, a[1010][6];
int ans;
cin >> n;
for (int i = 1; i <= n; i++) {
    cin >> a[i][0] >> a[i][1] >> a[i][2];
    a[i][3] = a[i][0] + a[i][1] + a[i][2];
}
for (int i = 1; i <= n; i++) {
for (int j = i + 1; j <= n; j++)
    if (abs(a[i][0] - a[j][0]) <= 5 &&
        abs(a[i][1] - a[j][1]) <= 5 &&
        abs(a[i][2] - a[j][2]) <= 5 &&
        abs(a[i][3] - a[j][3]) <= 10)
    ans++;
}
```

数组坐标的表示

一个格子的坐标值是 (x,y)，那么它是这个表格 x 行 y 列。

根据这个格子，可以推导出它附近格子的坐标，例如：

- 上方：x-1, y
- 下方：x+1, y
- 左方：x, y-1
- 右方：x, y+1

a[x][y] 是第 x 行第 y 列的数据

x-2,y-2	x-2,y-1	x-2,y	x-2,y+1	x-2,y+2
x-1,y-2	x-1,y-2	x-1,y	x-1,y+1	x-1,y+2
x,y-2	x,y-1	x,y	x,y+1	x,y+2
x+1,y-2	x+1,y-1	x+1,y	x+1,y+1	x+1,y+2
x+2,y-2	x+2,y-1	x+2,y	x+2,y+1	x+2,y+2

杨辉三角

例5.7（洛谷P5732）

下图是杨辉三角的前 6 行。

观察样例的规律，输出杨辉三角的前 n 行。

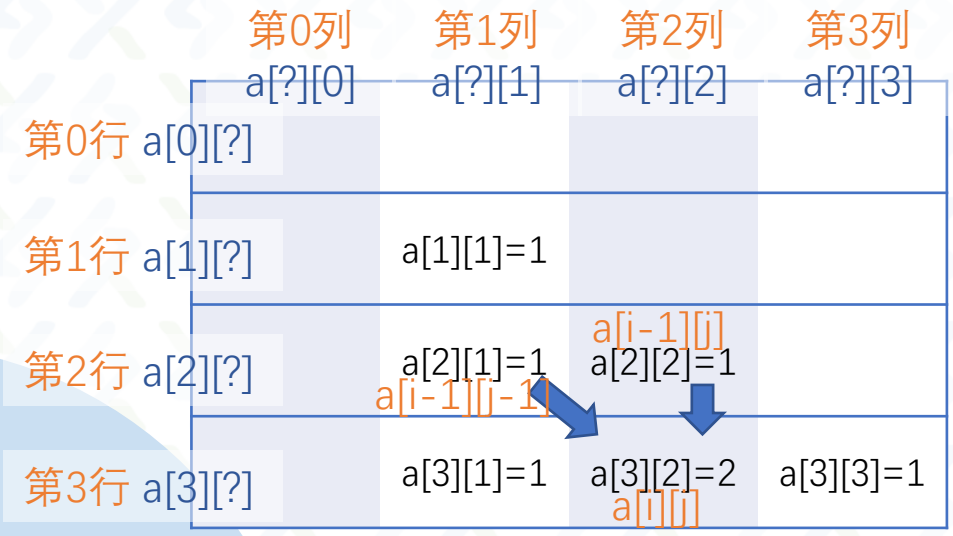
```
6
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

杨辉三角

杨辉三角的递推公式：a[i][j]= a[i-1][j] + a[i-1][j-1]。

注意循环边界条件，初始化时把 a[i][1] 和 a[i][i] 均赋为1。




```
#include<iostream>
using namespace std;
int a[21][21];
int main() {
  int n;cin >> n;
  for (int i = 1; i <= n; i++)
    a[i][1] = a[i][i] = 1; //赋初值
  for (int i = 1; i <= n; i++)
    for (int j = 2; j < i; j++)
  //a[i][1]、a[i][i]已赋值，故循环是2到n-1
      a[i][j] = a[i-1][j] + a[i-1][j-1];
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++)
      cout << a[i][j] << " ";
      cout << endl;
   }
}
```

洛谷

数组应用案例

本节将会介绍更多的关于数组的用法，题目会有一点复杂，前面的知识都会涉及，准备好了吗？

请翻至课本 P56

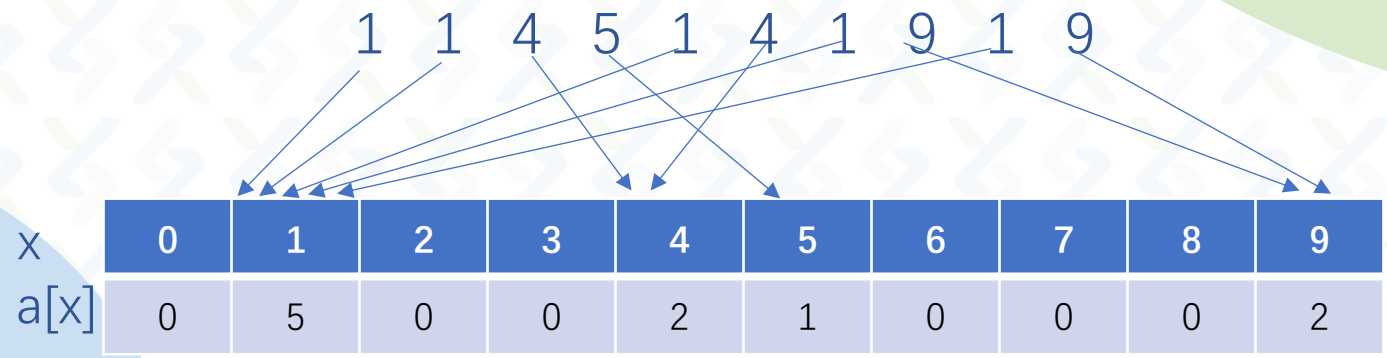
统计数字

例子

输入 n(n≤100) 和 n 个 0 到 10 的整数，然后输出 0 到 9 分别有多少个？

```
10
1 1 4 5 1 4 1 9 1 9
```

```
0 5 0 0 2 1 0 0 0 2
```



统计数字

设立数组 a[10]（注意只能用 a[0] 到 a[9]），分别放置数字 0 到 9 的数量。比如 a[3] 就是 3 个这个数字的数量。

需要初始化！每读入一个数字 x，就把 a[x] 增加 1。

最后输出 a 数组中的数字。这种做法叫做 桶计数。

```
#include<iostream>
using namespace std;
int n, a[10], x;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> x;
        a[x]++; // 用桶计数的方法把数存进去
    }
    for (int i = 0; i <= 9; i++) {
        cout << a[i] << " "; // 依次输出 0 到 9 的数字的数量
    }
    return 0;
}
```

彩票摇奖

例 5.8（洛谷 P2550）

每张彩票上印着 7 个各不相同的号码，取值范围为 1 到 33。

每次兑奖前公布七个各不相同的中奖号码。设置特等奖，一等奖至六等奖共七个奖项，兑奖时不考虑号码顺序

特等奖：要求彩票上 7 个号码都出现在中奖号码中；

一等奖：要求彩票上 6 个号码出现在中奖号码中；

二等奖：要求彩票上 5 个号码出现在中奖号码中；

以此类推。

现已知中奖号码和小明买的若干张彩票的号码，请你写一个程序帮助小明判断他买的彩票的中奖情况。

彩票摇奖

输入的第一行只有一个自然数 n，表示小明买的彩票张数；

第二行存放了 7 个介于 1 和 33 之间的自然数，表示中奖号码；

在随后的 n 行中每行都有 7 个介于 1 和 33 之间的自然数，分别表示小明所买的 n 张彩票。

依次输出买的彩票的中奖情况（中奖的张数），首先输出特等奖的中奖张数，然后依次输出一等奖至六等奖的中奖张数。

```
2
23 31 1 14 19 17 18
12 8 9 23 1 16 7
11 7 10 21 2 9 31
```

```
0 0 0 0 0 1 1
```

彩票摇奖

```cpp
#include <iostream>
using namespace std;
int main() {
    int n, a[10], num[10] = {0};
    cin >> n;
    for (int i = 1; i <= 7; i++)
        cin >> a[i]; // 首先使用数组 a 读入 7 个中奖号码。
    for (int k = 1; k <= n; k++) { // 循环 n 次读入每张彩票
        int ans = 0;
        for (int i = 1; i <= 7; i++) { // 读入每个彩票的数字
            int x; cin >> x;
            for (int j = 1; j <= 7; j++)
                // 每次比较每个号码是否为中奖号码
                if (a[j] == x) // 中了一个数字
                    ans++; // 中的数字增加 1
        }
        num[ans]++; // 中了 ans 个的数字的彩票数量增加 1
    }
    for (int  i = 7; i >= 1; i--)
        // 先输出中了 7 个数字的数量，然后 6 个……最后 1 个
        cout << num[i] << " " ; // 输出答案，加上空格
    return 0;
}
```

神奇的幻方

例 5.9（洛谷 P2615）

幻方是种 $N \times N$ 矩阵：里面填上 $1,2,\cdots\cdots,N \times N$，每行每列及两对角线上的数字之和都相同。现给定 N，请构造 $N \times N$ 的幻方。

当 N 为奇数时，可如此构建：首先将 1 写在第一行的中间；之后，按如下方式从小到大依次填写每个数 $K(K = 2,3,\cdots,N \times N)$：

- 若 (K-1) 在第一行但非最后列，将 K 填在最后行，(K-1) 所在列的右一列；
- 若 (K-1) 在最后列但非第一行，将 K 填在第一列，(K-1) 所在行的上一行；
- 若 (K-1) 在第一行最后一列，则将 K 填在 (K-1) 的正下方；
- 若 (K-1) 既不在第一行，也最后一列，如果 (K-1) 的右上方还未填数，则将 K 填在 (K-1) 的右上方，否则将 L 填在 (K-1) 的正下方。

神奇的幻方

使用一个二维数组 g 来存下这个幻方。

从 1 开始循环，依次按照构造规则判断属于哪一种情况。

根据不同的规则将数字填入到指定的位置（将题意翻译成代码），并且记录下刚才填写的坐标。最后把这个二维数组输出。

```
int n, g[40][40], x, y;
cin >> n;
g[1][n / 2 + 1] = 1;
x = 1; y = n / 2 + 1;
for (int i = 2; i <= n * n; i++) {
  if (x == 1 && y != n)
  // 第一行但不是最后一列
    g[n][y + 1] = i, x = n, y++;
  else if (y == n && x != 1)
  // 最后一列但不是第一行
    g[x - 1][1] = i, x--, y = 1;
  else if (x == 1 && y == n)
  // 第一行最后一列
    g[2][n] = i, x = 2;
```

```
  else if (x != 1 && y != n) {
  // 不在第一行，也不再最后一列
    if (g[x-1][y+1] == 0) //右上未填
      g[x - 1][y + 1] = i, x--, y++;
    else
      g[x + 1][y] = i, x++;
    continue;
  }
}
for (int i = 1; i <= n; i++) {
  for (int j = 1; j <= n; j++)
    cout << g[i][j] << " ";
  cout << endl;
}
```

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P78

校门外的树

判断练习

某校大门外长度为 L（不超过 10000）的马路上有一排间隔 1 米的树。

给出若干可能重叠区域，任一区域的起始和终止点的坐标都是整数。把这些区域中的树（包括端点）移走。最后马路上还有多少棵树。

请问这个程序有几处错误？

```
#include <iostream>
using namespace std;
int main() {
    int l, m, tree[10000], a,b,s=0;
    cin >> l >> m;
    for(int i = 0; i < m; i++) {
        cin >> a >> b;
        for(int j = a; j <= b; j++)
            tree[j] = 1;
    }
    for(int i = 0; i <= l; i++)
        if(tree[i] == 0) s++;
    cout << s << endl;
}
```

校门外的树

在应用数组时最容易犯的错误：

- 在函数内开数组，但是没有初始化（或读入）而直接使用。
- 数组数量没有开够。

```
#include <iostream>
// #include <cstring>
using namespace std;
int main() {
    int l, m,tree[10010]={0},a,b,s=0;
    cin >> l >> m;
    // memset(tree, 0, sizeof(tree));
    for(int i = 0; i < m; i++) {
        cin >> a >> b;
        for(int j = a; j <= b; j++)
            tree[j] = 1;
    }
    for(int i = 0; i <= l; i++)
        if(tree[i] == 0) s++;
    cout << s << endl;
    return 0;
}
```

总结

数组

数组是多个变量的集合

定义方式：数据类型 数组变量名称[元素个数]；

数组定义时应略微定义大一些，例如 int a[1010]

可以用 a[50]，a[i] 这种形式访问一个值

数组初始化

可用循环依次赋值，也可用memset或定义时初始化

注意初始化所花费的时间

不同题目初始化的值应分别考虑

总结

数组下标的应用

查找数组中的指定元素

在数组中插入/删除一个元素

多维数组

可定义多维数组，变成一个矩阵，例如 int a[105][105]

数组通常与循环结构共同使用

一些数组的比较复杂的应用

数组的坐标表示：(x,y) 坐标系统，往下 x 变大，往右 y 变大

桶计数，设置数组，用于统计每个元素的数量

课后拓展

习题 5.1 梦中的统计（洛谷 P1554）

Bessie 正在数数，她从 M 数到 N，请统计 M 到 N 中每个数码出现了几次？

例如 111 中 1 出现了 3 次，2-9 和 0 都出现了 0 次。

提示：可定义 a 数组，a[0] 表示0出现的次数，a[1] 表示 1 出现的次数...以此类推。分离每一个数字，得到一个数字就在相应的 a 数组上 +1。

这题依然需要初始化 a 数组。

课后拓展

习题 5.2 珠心算测验（洛谷 P2141 NOIP2014）

给出n（n<=100）个不超过 10000 互不相同的正整数，求这些数字中有多少个数恰好等于另外两个不同的数之和。

例如下面的例子，3=1+2，4=1+3，所以这两个数符合要求

提示：使用三重循环，第一重枚举判断各个数字，第二重和第三重枚举剩下两个数字。这三个数字不能在相同的位置。

```
4
1 2 3 4
```

```
2
```

课后拓展

习题 5.3 爱与愁的心痛（洛谷 P1614）

最近有 n（不超过 3000）个不爽的事，每句话都有一个正整数（不超过 100）刺痛值。请问连续 m 个刺痛值的和的最小值？

```
8 3
1 4 7 3 1 2 4 3
```

```
6
```

提示：枚举区间的头部（1 到 n-m+1），然后计算区间的和，然后和答案（ans）打擂台。

课后扩展

习题 5.6 蛇形方阵（洛谷 P5731）

输出一个如图 n×n 的蛇形方阵，每个字符占用 3 个宽度，使用“%3d” 输出。

```
4
```

```
  1   2   3   4
 12  13  14   5
 11  16  15   6
 10   9   8   7
```

课后拓展

习题 5.8 插火把（洛谷 P1789）

有一 n×n 的方阵现有 m 个火把和 k 个萤石，坐标已经知道。

它们的照亮范围入下图所示。

没有光或者没放东西的地方会生成怪物，请问方阵中有几个位置会生成怪物？

提示：火把位置 (i,j)，那么 (x,y) 满足 |x-i|+|y-i|<=2 会被照亮。

火把和萤石的照亮范围：

压缩技术

习题 5.9 压缩技术（洛谷 P1319）

设某汉字由 N × N 的 0 和 1 的点阵图案组成。N 不超过 200。

从汉字点阵图案的第一行第一个符号开始计算，按书写顺序从左到右，由上至下。

第一个数表示连续有几个0，第二个数表示接下来连续有几个1，第三个数再接下来连续有几个0，第四个数接着连续几个1，以此类推……现给出压缩码请求出点阵。

```
7 3 1 6 1 6 4 3 1 6 1 6 1 3 7
```

```
0001000
0001000
0001111
0001000
0001000
0001000
1111111
```

压缩技术

习题 5.10 压缩技术续集版（洛谷P1330）

设某汉字由 N × N 的 0 和 1 的点阵图案组成。N 不超过 200。

从汉字点阵图案的第一行第一个符号开始计算，按书写顺序从左到右，由上至下。

第一个数表示连续有几个0，第二个数表示接下来连续有几个1，第三个数再接下来连续有几个0，第四个数接着连续几个1，以此类推……现给出点阵请求出压缩码。

```
0001000
0001000
0001111
0001000
0001000
0001000
1111111
```

```
7 3 1 6 1 6 4 3 1 6 1 6 1 3 7
```

参考阅读材料

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

- P72 例 5.6：二位矩阵的循环枚举
- P73 例 5.7：超过二维的循环
- P76 例 5.10：使用数组存储字符，难度较大
- 习题 5.4、5.5、5.11。