CS 1502H 程序设计思想与方法 (C++)
Principles and Methodology in Programming
Chapter 5. 数组-2

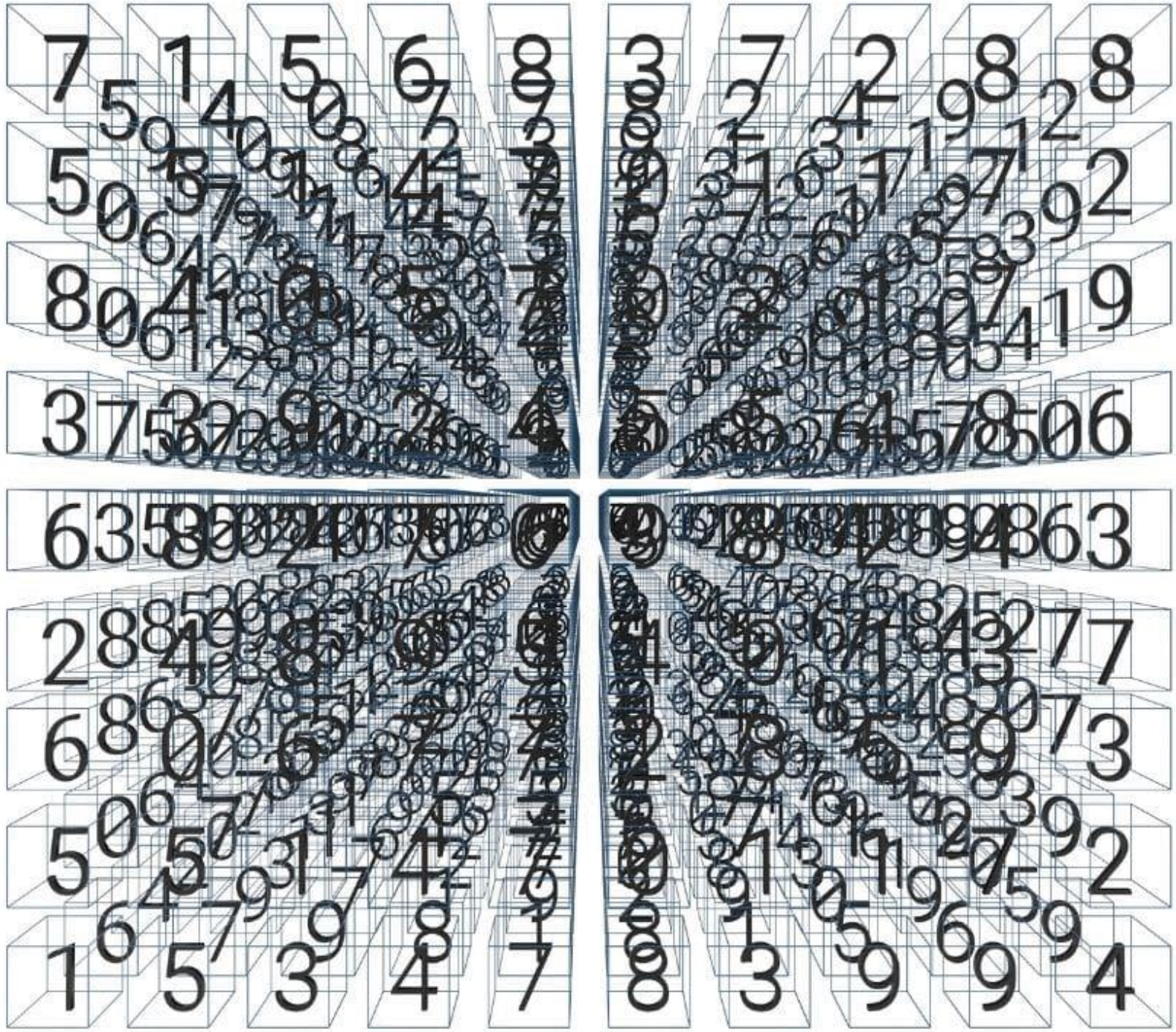
陈东尧
上海交通大学
2025年秋季

章节回顾

- 数组两大特性：同类型，内存中有序排放
- 数组的初始化、下标越界
- 二分查找算法
  - 二分查找的思路
  - 要求熟练掌握并能快速实现
- 排序算法

本章梗概

- 一维数组
- 排序和查找
- 二维数组
- 字符串



二维数组

- 数组的每一个元素又是数组的数组称为多维数组
- 最常用的多维数组是二维数组，又称为矩阵（matrix）
- 二维数组的定义格式：

  类型说明　数组名[常量表达式1][常量表达式2]

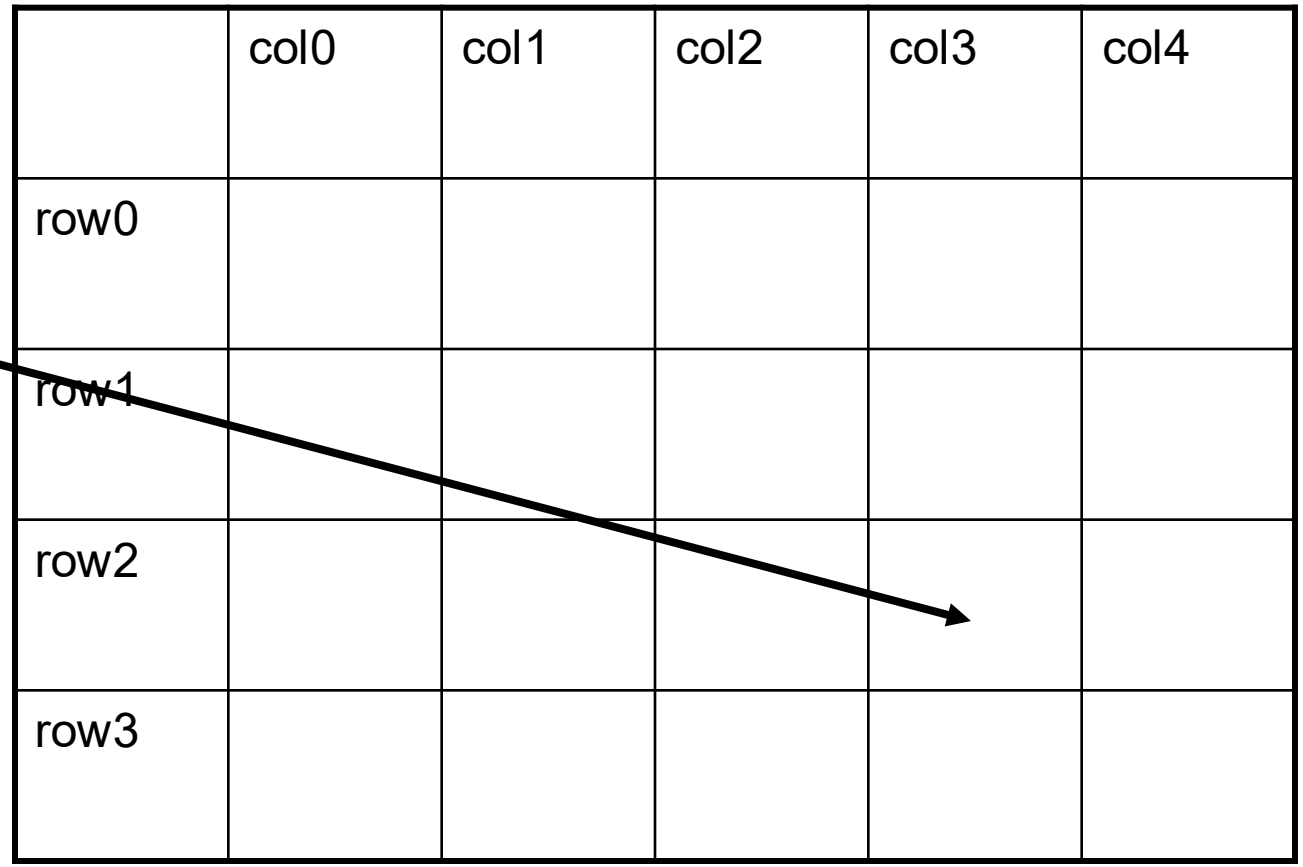
  常量表达式1表示行数，常量表达式2表示列数

  例如：int a[4][5]，相当于定义了20 个变量：

$$
\begin{matrix}
a[0][0], & a[0][1], & \ldots, & a[0][4] \\
\vdots & \ddots & & \vdots \\
a[3][0], & a[3][1], & \ldots, & a[3][4]
\end{matrix}
$$

二维数组

a[2][3]



	col0	col1	col2	col3	col4
row0					
row1					
row2					
row3					

二维数组初始化

- **格式：**

  类型说明 数组名[常量表达式1] [常量表达式2]={.....};

- **给所有的元素赋初值。如：**

  int a[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

  也可以通过{}把每一行括起来使这种初始化方法表示得更加清晰。

  int a[3][4] = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12}};

二维数组初始化

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **对部分元素赋值**

```
int a[3][4] = { 1, 2 , 3, 4, 5 };
```

- **为每一行的部分元素赋初值**

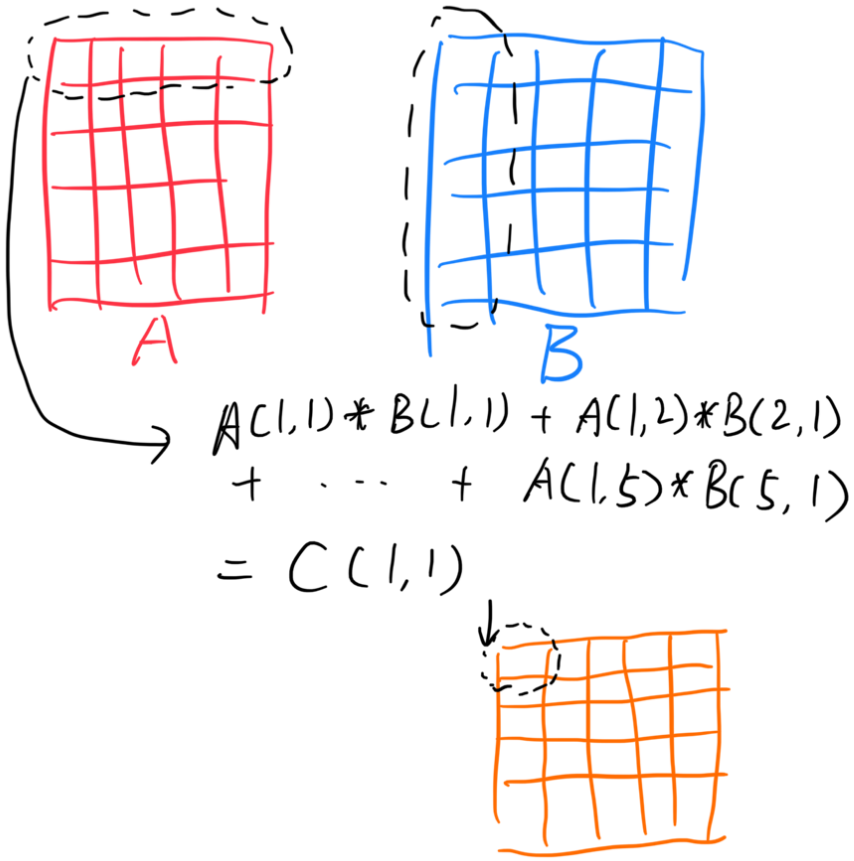
```
int a[3][4] = { {1, 2}, {3, 4}, {5}};
```

程序举例：矩阵乘法

矩阵乘法 C=A*B

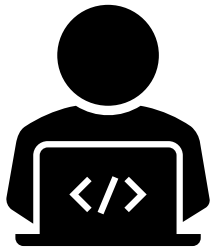
A[L][M]，B[M][N] ➔ C[L][N]

$$c[i][j] = \sum_{k=1}^{M} a[i][k] * b[k][j]$$



矩阵乘法

- 输入矩阵
- 执行乘法
- 输出结果



代码演示

chapter5_2d_array-cpp.cpp

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

int main(){
    int NumOfRowA, NumOfColA, NumOfColB;
    //Provide the metrics of A and B:
    cout<<"输入A的行数、列数和B的列数:";
    cin >> NumOfRowA >> NumOfColA >>NumOfColB;
    int a[NumOfRowA][NumOfColA], b[NumOfColA][NumOfColB], c[NumOfRowA][NumOfColB];

    cout<<"\n输入数组A:\n";
    for (int i = 0; i < NumOfRowA; i++){
        for (int j = 0; j < NumOfColA; j++){
            cout<<"a["<<i << "][" << j << "] = ";
            cin >> a[i][j];
        }
    }
    cout<<"\n输入数组B:\n";
    for (int i = 0; i < NumOfColA; i++){
        for (int j = 0; j < NumOfColB; j++){
            cout<<"b["<<i << "][" << j << "] = ";
            cin >> b[i][j];
        }
    }
    for (int i = 0; i < NumOfRowA; i++){
        for (int j = 0; j < NumOfColB; ++j){
            c[i][j] = 0;
            for (int k = 0; k < NumOfColA; ++k){
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    cout<<"\n输出数组C:";
    for (int i = 0; i < NumOfRowA; i++){
        cout<<endl;
        for (int j = 0; j < NumOfColB; j++){
            cout<<c[i][j]<<'\t';
        }
    }
}
```

矩阵乘法

注：
课程资料中提到数组的定义需要用常量表达式。然而较新型的c++编译器（如gcc）的编译过程是可以通过这种初始化方法的：

int a[NumOfRowA][NumOfColA]

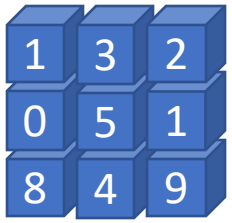
需要强调的是：C++的官方标准是不允许用变量来作为数组长度的。C++的标准草案中也明确了这一点[1]。代码能运行只是因为编译器比如gcc提供了扩展功能。
所以大家在程序实践时这样操作是可以的。**答题时还请以书本上为准**，即在C++代码中尽可能不用变量来定义数组长度。
更好的方法是利用const int MAX_SIZE来预定义数组的最大规模[2]，或者利用后续学到的malloc函数和new操作符来动态分配。

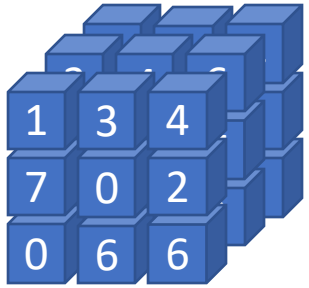
[1] https://eel.is/c++draft/dcl.array#1
[2] https://en.cppreference.com/w/c/language/array
https://copyprogramming.com/howto/why-no-variable-size-array-in-stack

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

int main(){
    int NumOfRowA, NumOfColA, NumOfColB;
    //Provide the metrics of A and B:
    cout<<"输入A的行数、列数和B的列数:";
    cin >> NumOfRowA >> NumOfColA >>NumOfColB;
    int a[NumOfRowA][NumOfColA], b[NumOfColA][NumOfColB], c[NumOfRowA][NumOfColB];

    cout<<"\n输入数组A:\n";
    for (int i = 0; i < NumOfRowA; i++){
        for (int j = 0; j < NumOfColA; j++){
            cout<<"a["<<i << "][" << j << "] = ";
            cin >> a[i][j];
        }
    }
    cout<<"\n输入数组B:\n";
    for (int i = 0; i < NumOfColA; i++){
        for (int j = 0; j < NumOfColB; j++){
            cout<<"b["<<i << "][" << j << "] = ";
            cin >> b[i][j];
        }
    }
    for (int i = 0; i < NumOfRowA; i++){
        for (int j = 0; j < NumOfColB; ++j){
            c[i][j] = 0;
            for (int k = 0; k < NumOfColA; ++k){
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    cout<<"\n输出数组C:";
    for (int i = 0; i < NumOfRowA; i++){
        cout<<endl;
        for (int j = 0; j < NumOfColB; j++){
            cout<<c[i][j]<<'\t';
        }
    }
}
```

扩展阅读：标量，向量，矩阵，张量

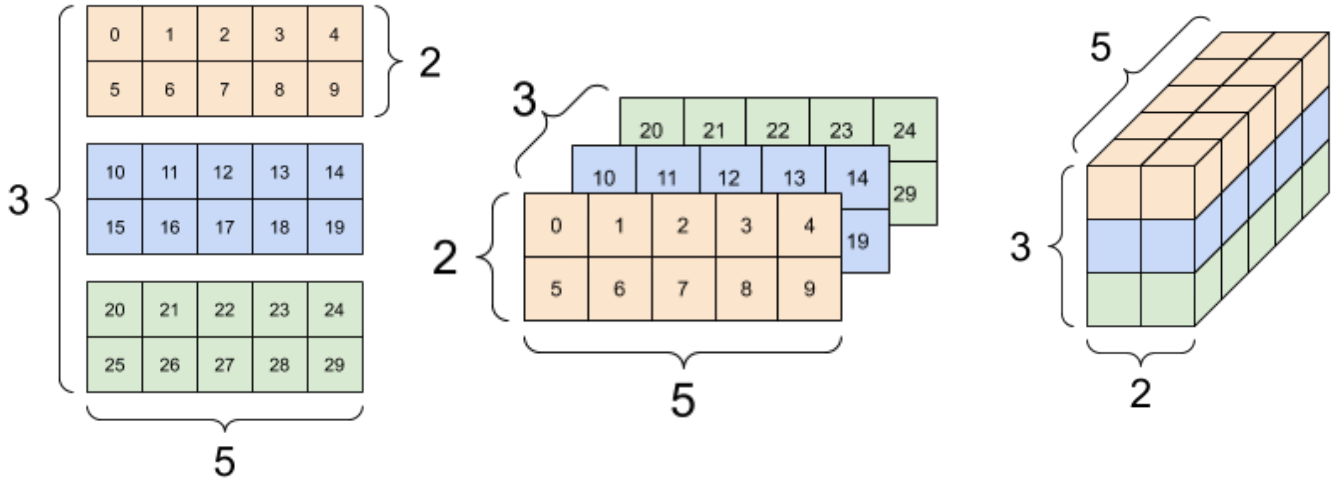
- 标量（scalar），向量（vector），矩阵（matrix），张量（tensor）



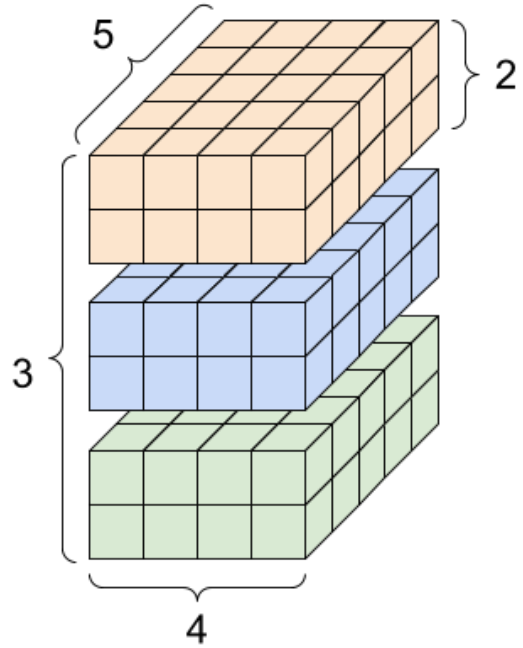


张量有多种表示法

- 3阶张量

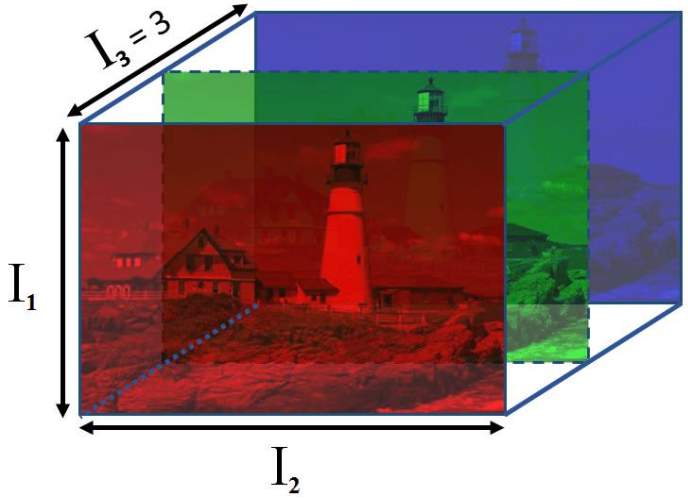


- 4阶张量



https://www.tensorflow.org/guide/tensor

标量，向量，矩阵，以及张量的应用



RGB图像即是3阶（order）的张量

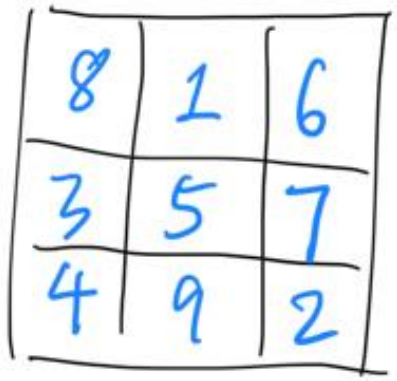
TensorFlow

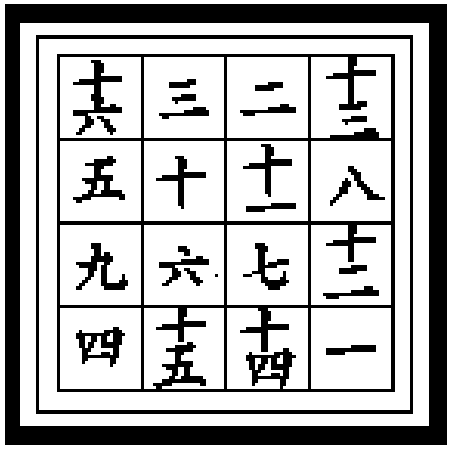
PyTorch

基于张量的计算流程也是机器学习**的基**础框架

程序举例：输出魔术方阵（Magic square）

- 魔术方阵（又称纵横图）：是指将1~n*n个正整数排列成n x n阶方阵，n为奇数。它具有这样的性质：每行，每列，以及对角线之和都是相等的，都是**n(n*n+1)/2**
- 如何设计一个程序，用户只需定义方阵的阶（order），程序会打印出对应的魔术方阵。





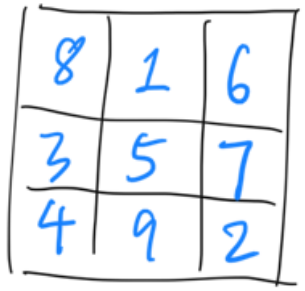
16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

四阶纵横图

杨辉在《续古摘奇异法》（1275）卷一始有“纵横图”之名，其中给出了三至十阶的幻方及其变体共十三种

程序举例：输出魔术方阵（Magic square）

- 第一个问题，魔术方阵如何构建？



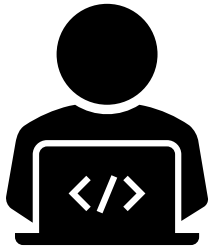
$\frac{3\times(3^2+1)}{2}=\frac{3\times10}{2}=15$

"↱" OR "↓"

1. 第一行中间的元素填“1”
2. 向上走一格，向右再走一格，在此处填下一个数。
   1. 如果行列范围超出矩阵范围，则**回绕**（例如1在第1行，则2应放在最下一行，列数同样加1）
   2. 如果按上面规则确定的位置上已有数，或上一个数是第1行第n列时，则把下一个数放在上一个数的下面
   3. 所有格子里都有数后，完成

程序写法

- 注意魔术方阵的填写规则



代码演示

chapter5-magic-square-1.cpp
chapter5-magic-square-2.cpp

```
#define MAX 100 //最高位100阶

int main(){
    int scale = 0;
    cin >> scale;
    // vector< vector<int> > mat(scale, vector<int>(scale));
    int mat[MAX][MAX] = {0}; // 报错：  variable-sized object may not be initialized

    int numElement = 1, row=0, col = scale/2, tmpRow = 0, tmpCol = 0;
    mat[row][col] = numElement;
    while(numElement++ < scale*scale){
        // 观察下一步走法：
        tmpRow = row-1, tmpCol = col+1;
        if(tmpRow < 0) tmpRow = scale-1;
        if(tmpCol == scale) tmpCol = 0;
        // 确定下一步位置：
        if(mat[tmpRow][tmpCol] != 0) {
            row = row+1;
            if(row == scale) row = 0;
            }else{
            row = tmpRow;
            col = tmpCol;
            }
            //填数：
            mat[row][col] = numElement;
            }

            for (int i = 0; i < scale; i++){
            for (int j = 0; j < scale; j++){
            cout<<mat[i][j]<<'\t';
            if (j == scale-1) cout<<'\n';
        }
    }
}
```

本实例中常见的错误预警

- 用变量初始化数组报错：variable-sized object may not be initialized

```
magic-test.cpp:10:13: error: variable-sized object may not be initialized
    int mat[scale][scale] = {0};
            ^~~~~
```

- 存储器区段错误（中文简称段错误）：Segmentation Fault

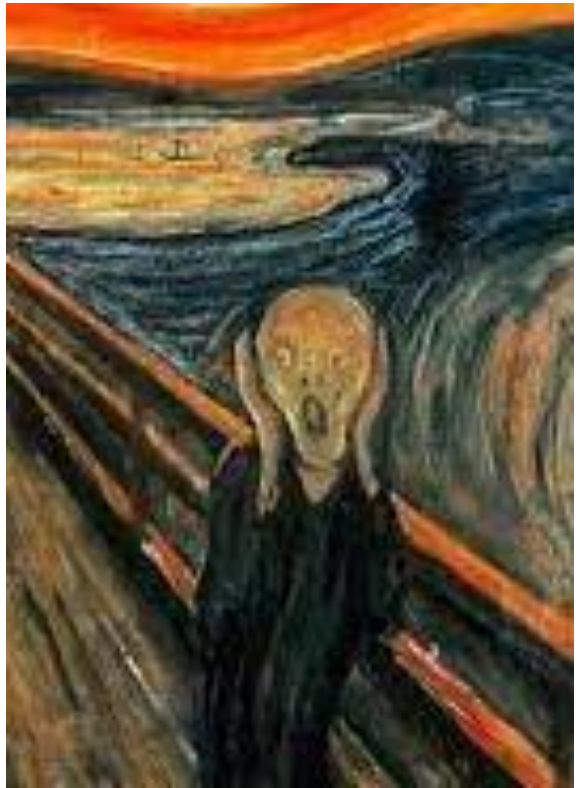
```
t.cpp -o magic-test && "/Users/chendy/Documents/Teaching/CppClass/5-数组/"magic-test
3
zsh: segmentation fault  "/Users/chendy/Documents/Teaching/CppClass/5-数组/"magic-test
```

简称SegFault

关于段错误（Segmentation Fault）

尝试读取程序地址空间之外的数据或写入至只读内存段时，会导致储存器段错误

编译器的提示对于SegFault的提示非常少，需要设计者的仔细分析



关于段错误（Segmentation Fault）

尝试读取程序地址空间之外的数据或写入至只读内存段时，会导致储存器段错误

- 当程序试图访问不允许访问的内存位置，或试图以不允许的方式访问内存位置（例如，**尝试写入只读位置**，或覆盖操作系统的一部分）时，会产生储存器段错误。例如数组访问越界可能产生这个错误
- 术语“分段”在计算中有多种用途；“储存器段错误”是自1950年代以来就一直使用的术语。当有**内存保护**时，只有程序自己的地址空间是可读的，其中只有堆栈和程序数据段的可读写部分是可写的，而只读数据和代码段是不可写的。因此，尝试读取程序地址空间之外的数据或写入至只读内存段时，会导致储存器段错误。

【扩展阅读】Segmentation fault产生的原因和讲解
https://gywbd.github.io/posts/2016/1/segmentation-fault.html

关于段错误（Segmentation Fault）

- 如何避免、检查segfault：
  - 检查变量是否正确初始化
  - 检查循环变量是否越界
  - 检查<或<=是否使用正确

预告：**指**针相关的代码更容易遇到segfault 

本章梗概

- 一维数组
- 排序和查找
- 二维数组
- 字符串

字符串

- 由一系列字符组成的一个数据称为字符串
- 在C++中，字符串常量用一对双引号括起来。如“Hello,world”
- 字符串变量：用字符类型的数组来表示
- 字符串的本质是一系列的有序字符，因此可以用一个字符数组来保存这组字符 。用数组名表示这个字符串

字符串存储

- 由于数组名是数组的起始地址，因此该字符串从该地址开始存储。但到哪里为止？
  - C++用‘¥0’表示字符串的结束。
  - 字符串所需的存储空间比实际的字符串长度大1
  - 如要将字符串”Hello,world”保存在一个数组中，该数组的长度为

    12

字符串初始化

- **char ch[ ] = { ‘H’, ’e’, ’l’, ’l’, ’o’, ’,’, ’w’, ’o’, ’r’, ’l’, ’d’, ’¥0’};**
- **char ch[ ] = {“Hello,world”};**
- **char ch [ ] = “Hello,world”;**

空字符串

- 不包含任何字符的字符串称为空字符串。
- 空字符串占用的空间为1个字节，存储‘¥0’
- 注意‘a’和“a”的区别
  - ‘a’是简单变量
  - “a”是数组

字符串的输入输出

- **逐个字符的输入输出：这种做法和普通的数组操作一样。**
  - **将整个字符串一次性地用cin和cout输入或输出。**

cin和cout

如定义了一个字符数组ch。要输入一个字符串放在ch中，可直接用

cin >> ch;

要输出ch的内容。可直接用

cout << ch;

注意：cin输入是以空格、回车或Tab键作为结束符。因此无法输入包含空白字符的字符串在用cin输入时，要注意输入的字符串的长度不能超过数组的长度。因此，最好在输出的提示信息中告知允许的最长字符串长度。

【扩展阅读】C++ getline介绍文档
https://www.cplusplus.com/reference/string/string/getline/

getline（）函数

如果要读取包含空格的一句话
cin.getline(字符数组，数组长度，结束标记)

- 读取包含任意字符的字符串
- 直到达到结束标记或者字符串读满（包含¥0）

```
char name[256];
cout << "Please input your name: ;
cin.getline(name, 256);
```

类似的还有gets()

【扩展阅读】C++ gets介绍文档
https://www.programiz.com/cpp-programming/library-function/cstdio/gets

字符串处理函数(重点！)

- 字符串不能直接用系统的内置运算符进行操作
- C++的函数库中提供了一些用来处理字符串的函数。这些函数在库cstring中

函数	作用
strcpy(dst, src)	将字符从 **src**拷贝到**dst**。函数的返回值是**dst**的地址
strncpy(dst, src, n)	至多从 **src** 拷贝**n**个字符到**dst**。函数的返回值是**dst**的地址
strcat(dst, src)	将 **src** 接到 **dst** 后。函数的返回值是**dst**的地址
strncat(dst, src, n)	从 **src** 至多取 **n** 个字符接到 **dst** 后。函数的返回值是**dst**的地址
strlen(s)	返回**s**的长度
strcmp(s1, s2)	比较 **s1** 和 **s2**。如 **s1 > s2** 返回值为正数，**s1=s2**返回值为**0**，**s1<s2**返回值为负数
strncmp(s1, s2, n)	如 **strcmp**，但至多比较**n**个字符
strchr(s, ch)	返回一个指向**s**中第一次出现**ch**的地址
strrchr(s, ch)	返回一个指向**s**中最后一次出现**ch**的地址
strstr(s1, s2)	返回一个指向**s1**中第一次出现**s2**的地址

函数原型：char * strcpy (char * destination, const char * source);
//讲完函数和指针后可以完全理解此函数原型的含义

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

int main ()
{
      char str1[]="Sample string";
      char str2[40];
      char str3[40];
      strcpy (str2,str1);
      strcpy (str3,"copy successful");
      // printf ("str1: %s\nstr2: %s\nstr3: %s\n",str1,str2,str3);
      cout<<str1<<endl<<str2<<endl<<str3<<endl;
      return 0;
}
```

```
Sample string
Sample string
copy successful
```

int strcmp (const char * str1, const char * str2);
//讲完函数和指针后可以完全理解此函数原型的含义

return value	indicates
<0	the first character that does not match has a lower value in ***ptr1*** than in ***ptr2***
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in ***ptr1*** than in ***ptr2***

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

int main() {
char str1[] = "Mega";
char str2[] = "Meta";

// compare str1 and str2 lexicographically
int result = strcmp(str1, str2);

cout << result;

return 0;
}
```

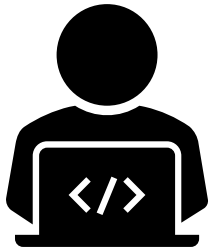
-1

字符串的应用

- 输入一行文字，统计有多少个单词。单词和单词之间用空格分开。
- 解题关键：单词的数目可以由单词间的空格决定
- 解题思路：
  - 设置一个计数器num表示单词个数。开始时，num=0。
  - 从头到尾扫描字符串。当发现当前字符为非空格，而当前字符以前的字符是空格，则表示找到了一个新的单词，num加1。
  - 当整个字符串扫描结束后，num中的值就是单词数。

字符串的应用

- **输入一行文字，统计有多少个单词。单词和单词之间用空格分开。**
- **注意，文字中不包括标点符号。**



代码演示

chapter5-word-count.cpp

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

int main() {
char sentence[80], prev = ' '; //prev 表示当前字符的前一字符
int i, num = 0;

cin.getline(sentence, 80);

for (i = 0; sentence[i] != '\0'; ++i) {
        if (prev == ' ' && sentence[i] != ' ') ++num;
        prev = sentence[i];
}
cout << "单词个数为： " << num << endl;
return 0;
}
```

总结

- 数组通常用来存储具有同一数据类型并且按顺序排列的一系列数据。数组中的每一个值称作元素，通常用下标值表示它在数组中的位置。**在C++中，下标是从0开始的。**
- 定义一个数组时，必须定义数组的大小，而且**它必须是常量**
- 数组中的元素一般用数组名后加用方括号括起来的下标表示
- 数组元素通常是连续存储的。第一个元素的地址称为基地址
- 数组的下标可以是任意的计算结果可以自动转换成整型数的表达式，包括整型，字符型或者枚举型。
- 可以定义数组是多维的。多维数组可以看成数组的数组。