# 数组

数组是用来存储一系列相同类型的数据集合。程序把他们放在同一块连续的内存空间中，方便用户访问。

# 数组声明注意：

1. 存储在每个元素中的值的类型
2. 数组名称
3. 数组必须给予大小

# 数组声明

最简单的

数据类型 数组名称[数组元素个数]；

**注意：数组在进行创建时，必须给予有效的初始化**

# 数组的初始化方式：

float ValueArray[3] = {1.0, 2.0, 3.0};

float ValueArray[3]{1.0, 2.0, 3.0};

float ValueArray[]{1.0, 2.0, 3.0}; //自动推断数组长度

float ValueArray[] = {1.0, 2.0, 3.0};//自动推断数组长度

注意：

float ValueArray[5]{}//所谓位置为0

float ValueArray[5]{1.0， 2.0}//出去0，1位，其他位置为0

当我们进行数组数据填充时，填充数据量小于数组给定长度，则空白位置将用0进行填充。

# 数组访问

数组中存储的数据是连续的，需要使用下标来进行访问。注意数组的下标是从0开始的，也就是第一位数据位下标为0，最后一位数据的下标是数组长度减去1。

# C++11数组初始化

* 省略等于号初始化

float ValueArray[5];//不建议使用此初始化方式，装填数据不明确

* 大括号内填入数据不满，空白处用零初始化

float ValueArray[5]{1.0};//第一位为1.0，其余位置为0

* 大括号留白将所有元素设置为零

float ValueArray[5]{};//所有元素位为零

float ValueArray[5] = {};//同上

* 列表中禁止将数据向下扩容，缩窄转换

int ValueArray[5]{1.0};//禁止将浮点型转换到int整形进行填充

# 数组长度获取

Sizeof（arrayname） / sizeof（arrytypename） **只适用于普通数组不适用于指针数组**

C++本身不提供数组边界检查，操作时需要注意数组越界问题

# 字符串

用来记录一连串字符集合的数据结构，我们称之为字符串结构

char Name[] = { 'a', 'b', 'c', 'd', 'e' };//字符数组

char Name\_01[] = { 'a', 'b', 'c', 'd', 'e', '\0'};//字符串

在C++中约定了一下，当字符数组的最后一位是转义符\0时，我们将当前的字符数组认定为是字符串

# 常用转义符

\n 换行符号

\t 水平制表符

\b 退格

\r 回车

\” 双引号转义符，专门用来在字符串中进行双引号输入使用（只是针对英文双引号的转义操作使用）

# C语言风格字符串

char Name[11] = "哈喽";//c语言风格的字符串构建方式

char Name[] = "哈";//c语言风格的字符串构建方式 并且数组的大小是编译器计算的

# Cin对象的使用

Cin对象是iostream提供给我们的供用户进行输入信息收集的对象，我们可以通过cin对象从控制台收集用户输入信息，进行计算操作。Cin遇到回车或者空格则停止收集。

### Getline函数

Cin中的getline函数可以解决在输入信息中存在空格被截断的问题，这样可以更方便将内容进行输出。Getline函数按照整行数据进行获取，然后通过获取数量填入到字符数组中。切记获取数量禁止超过数组长度。

*cin*.*getline*(Name, 11);//遇到回车进行收集，收集整行内容

# String

C++98中加入的string类

特点：动态调整空间大小，而不受制于声明空间的大小

### String的使用=

**必须引入头文件#include <string>**

创建变量

string Name;//无赋值初始化

string Name = “OK”;//c风格字符串进行初始化

借助字符数组进行初始化

char Value[] = “OK”;

string Name = Value;//将字符数组中的内容填充到string对象中

### 获取整行输入

*string* Name; //构建的是一个变量，是一个string类型的变量

//cin >> Name;

*getline*(*cin*, Name);//根据输入行进行内容获取

### C++11中初始化string对象的方法（两种）

*string* Name = {"ok"};

*string* Name {"ok"};

### 字符串的拼接组合

string支持运算符+、+=的使用（重载了运算符操作），我们可以将两个字符串进行拼接

拼接格式：一下格式均对

*string* Name = {"ok"}; //构建的是一个变量，是一个string类型的变量

*string* time = "hello";

///////

*string* t1 = Name + time;//t1是 hellook

//////

time += Name;//time是 hellook

//////

支持直接与c风格字符串进行拼接运算

time += "OK";//被允许

time = Name + "OK";//被允许

# 数组的维度

一维数组：数组操作下标只有一个，只有单向操作。

二维数组：数组中操作下标有两个，二维数组可以理解为将一维数组进行了堆叠。

### 二维数组的构建初始化

int Data[3][3] = {};//构建所有元素都是0的二维数组，

int Data[3][3] = {

{3，8，8},//[0][n]

{},//[1][n]

{},//[2][n]

};//构建有效数据填充的二维数组

自动计算数组维度

int Data[][3] = {

{1, 3},//[0][n]

{},//[1][n]

{},//[2][n]

};

编译器将会自动计算高纬度数组的长度（这里就是3），千万注意，低维度禁止数组使用自动计算，也就是留白。

# For循环

流程控制语句，可以改变程序的执行流程，把程序锁定到一段逻辑中，通过给定的条件，进行任务的执行，达到循环操作的目的

遍历：通常指把数组结构中的数据进行逐一查询

For结构

For（初始化语句，终结条件语句，逻辑语句）

初始化语句一般用来初始化循环终结条件控制变量，循环过程中只执行一次

条件语句，一般用来控制逻辑终结条件，循环过程中每次都会进行检查

逻辑语句，一般用来更新控制变量，循环过程中每次都会执行。

### 前缀和后缀运算格式

操作基本数据时，递增+1，或是-1，指针操作除外

i++先进行操作（例如赋值），后进行运算+1

++i先进行运算+1，在进行操作