## 数组

1. 数组维度必须是常量表达式。

unsigned int N **=** 42**;**

int a**[**N**];**//错误，N不是常量表达式。常量表达式的意思是：数值不变，且在编译期值可知。在g++中，上述语句正常工作，可能原因是编译器扩展，也有说c++标准c99支持VLA（variable length array）

1. 四不能：不能使用auto自动推断数组元素类型，数组元素不能是引用类型，数组不能拷贝，数组不能赋值。

int array**[**2**]** **=** **{**1**,**2**};** //正确

auto a**[**2**]** **=** **{**1**,**2**};** //错误，数组元素类型不能使用auto自动推断

int **&** b**[**2**]** **=** /\*\*/ //错误，数组元素类型不能是引用

int c**[]** **=** array**;** //错误，数组不能拷贝

c **=** array**;** //错误，数组不能直接赋值

1. 使用auto获得的数组类型是指针类型；使用decltype获得数组类型。

auto a **=** array；//a的类型是指针类型

decltype（array）a **=** **{**1，2**}**；//decltype（array）是数组类型

1. begin（），end（）函数，头文件iterator中。
2. 数组索引类型是int 而不是unsigned int，索引可以为负值。

int array**[]** **=** **{**1**,**2**,**3**,**4**,**5**};**

int **\***p **=** **&**array**[**2**];**

int k **=** p**[-**2**]**//正确，表示array[0]

1. c风格字符串

const char **\*** a**[]** **=** “aaa”**;**

const char**\*** b**[]=**”bbb”**;**

比较：strcmp，若使用==比较的是指针值

连接：strcat，若使用+，表示两个指针相加，错误

1. 旧接口

string s **=** “sss”**;**

const char**\*** a**[]** **=** s**.**c\_str**();**//a[4] = “sss\0”;

1. 多维数组

除了最后一层范围for循环，其他for循环必须使用引用。

int a**[**2**][**2**]** **=** **{{**1**,**1**,},{**2**,**2**}};**

**for(**auto a1**:** a**){**

**for(**auto a2 **:** a1**){**//错误，因为a1退化为指针，类型为 int\* ，对int\*遍历是错误的

cout**<<**a2**<<**endl**;**

**}**

**}**

int a**[**2**][**2**]** **=** **{{**1**,**1**,},{**2**,**2**}};**

**for(**auto**&** a1**:** a**){**

**for(**auto a2 **:** a1**){**//正确，a1类型为int [2]

cout**<<**a2**<<**endl**;**

**}**

**}**

## 表达式

1. 优先级、结合性、求值顺序

求值顺序确定的只有逻辑与（&&），逻辑或（||），逗号运算符（，），双目运算符（？：），其他运算符不保证运算顺序：

a**+**b**\***c**+**d**;**//优先级保证先算b\*c，结合性保证接下来计算a+然后+d

cout**<<**i**<<++**i**<<**endl**;**//不保证先做++i还是先输出i，结果未定义

f**()+**g**();** //不保证先计算f()；如果f（）和g（）由共用变量，结构未定义

1. typedef

**typedef** int int\_array**[**4**];**//表示int\_array是数组元素类型为int，维度为4

int\_array a**;**//等价与int a[4];

**typedef** char**\*** P**;**

const char**\*** p1**;**//等价与char const \*p1;底层const

const P p2**;**//等价与 P const p2; 即就是 char\* const p2;顶层const

1. decltype（X）当X为左值时，得到类型为引用类型

int **\*** p **;**

decltype**(\***p**)** a**;**//因为解引用得到的是左值，所以 a 的类型是 int&

decltype**(&**p**)** b**;**//取值符 & 获得右值，所以得到 b 的类型是 int\*\*

1. 算术运算符

|  |
| --- |
| 一元正号 +  一元负号 - |
| 乘法运算符 \*  除法运算符 /  取模运算符 % |
| 二元加法 +  二元减法 - |

* 1. 运算结果都为右值
  2. 一元正号、加法、减法可用于指针
  3. 取模运算符 % 操作数都为整数，且符号和操作符左边的数相同：

21 **%** **-**5 **=** 1； 21**/-**5**=-**4；//必须保证 m = （m/n）\*n + m%n = m ，xain't所以有余数符号和操作符左边数相同

**-**21 **%** 5 **=** **-**1； **-**21**/**5**=-**4；

* 1. 整数除法向0取整：

**-**1**/**3**=**0；

**-**4.3**=-**1；

1**/**3**=**0；

1. 关系运算符

|  |
| --- |
| ！ |
| >  >=  <  <= |
| ==  != |
| && |
| || |

只有？：、&&、||、，这四种运算符保证求值顺序

1. 赋值运算符

赋值运算符优先级较低，复合赋值运算符和赋值运算符之间有差异：

a **+=** b**;**//a取值一次

a **=** a **+** b**;**//a取值两次

1. 位运算符

|  |
| --- |
| 求反 ~ |
| <<  >> |
| & |
| 异或^ |
| | |

1. 整型提升
2. 有符号右移未定义，一般来说用于无符号
3. int只保证16位，long保证32位
4. 优先级小于算术运算符，大于关系运算符、赋值运算符、条件运算符

左结合性

1. 条件运算符

只有当两个表达式都为左值时，结果为左值。

1. sizeof
   1. 优先级和解引用\*等同

右结合性

* 1. 优先级等同与解引用\*，右结合律

int **\***p **=** **nullptr;**

**sizeof** **\***p**;**//右结合性兵器优先级等同于 \* ，不对表达式求值所以正确

* 1. c++11新特性允许使用域运算符获取成员大小

class ClassA**{**

public**:**

int a**;**

**}**

**sizeof(**ClassA**::**a**);**//正确，一般来说访问类成员需要类对象，sizeof无须我们提供一个对象

1. 类型转换
   1. 隐式转换：
      1. 算术提升

char a**;**

short b**;**

a**+**b**;**//a 和 b 都转换为 int

* + 1. 指针和数组

int a**[**2**]** **=** **{**2**,**1**};**

int f**(**int**\*** p**){...};**

f**(**a**);**//a 转换为指针

* + 1. 指针

int**\*** pi**;**

void**\*** pv **=** pi**;**//int\* pi转换为 void\* pv

* + 1. bool

**while(**cin**>>**a**){**//istream对象转换为bool类型

/\*...\*/

**}**

* + 1. const

int a **=** 8**;**

const int b **=** a**;**//int 类型 隐式转换为 const int

* + 1. 类类型定义

string s **=** "hello world"**;**//const char\* 隐式转换为 string

派生类转换基类

* 1. 显式转换
     1. static\_cast<type>()：任何具有明确定义的类型转换，并且没有底层const
     2. const\_cast<type>()：去掉const属性
     3. reinterept\_cast<type>()：位模式下对对象重新解释

int a **=** 8**;**

char**\*** p **=** **reinterpret\_cast<**char**\*>(&**a**);**

* + 1. dynamic\_cast<type>()：派生类转换为基类类型