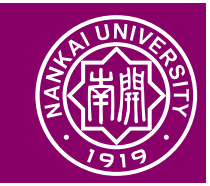


第二章 数据类型

主讲：刘晓光
张海威 张莹
殷爱茹 李雨森
宋春瑶 沈玮
卢少平

南開大學

- 数据与数据表示
- 基本数据类型
- 复合数据类型
- CV限定数据类型
- 说明语句

数据与数据表示

基本数据类型

复合数据类型

CV限定数据类型

说明语句

数据

什么是数据

- 计算机处理的对象
- 以某种特定的形式存在（数据类型）
- 数据之间存在某些联系
- 数据在程序设计中的地位
  - 第一要素
- 程序设计中，数据用常量和变量进行描述
  - 数组元素、类对象的性质与变量类似

数据类型

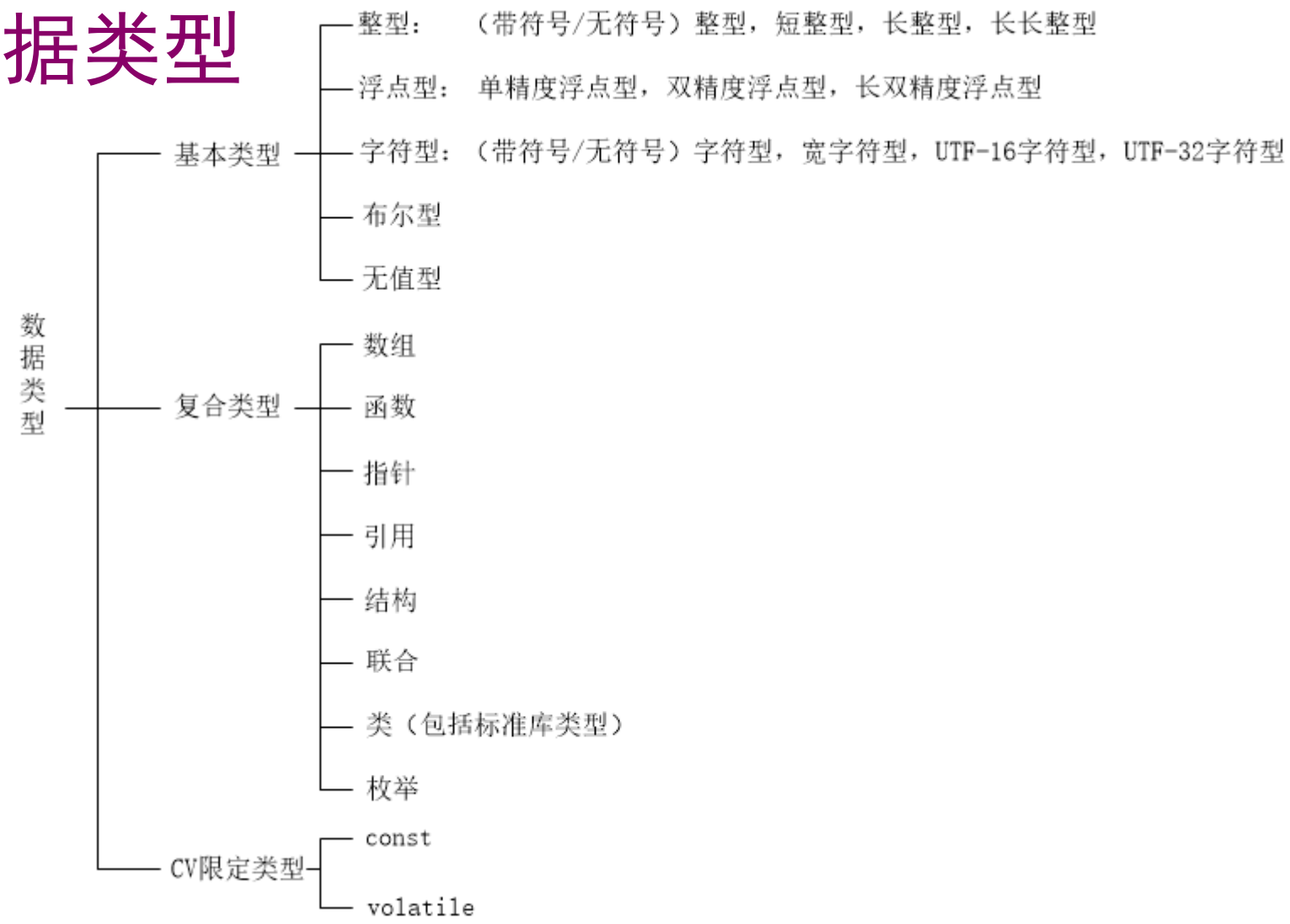
每一项数据应唯一地属于某种类型

每一数据类型意味着一个有明确定义的值的集合

同一类型的数据占有相同大小的存储空间

同一类型的数据具有相同的（允许对其施加的）运算操作集

数据类型


数据类型
基本类型
整型： （带符号/无符号）整型，短整型，长整型，长长整型
浮点型： 单精度浮点型，双精度浮点型，长双精度浮点型
字符型： （带符号/无符号）字符型，宽字符型，UTF-16字符型，UTF-32字符型
布尔型
无值型
复合类型
数组
函数
指针
引用
结构
联合
类（包括标准库类型）
枚举
CV限定类型
const
volatile

常量

程序运行过程中不能改变值的数据，分为

- 字面值常量
  - 整型字面值常量、浮点型字面值常量、字符型常量、字符串字面值常量、用户定义字面值常量……
- 常变量
  - 用const对变量进行限定
  - 说明常变量
    - const <数据类型> <常量名> = <表达式>
- 宏定义常量
  - #define <宏名> <宏替换体>
    - 宏名可理解为常量名，宏替换体可理解为常量值

变量

变量

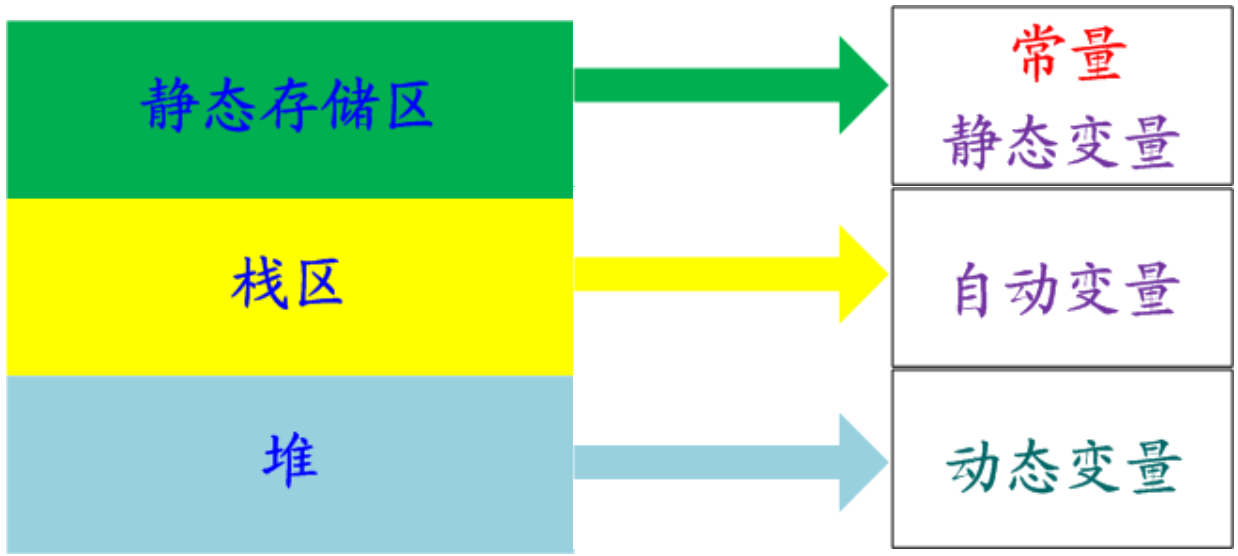
- 程序运行过程中其值能够改变的数据
- 根据变量的数据类型，在内存中开辟相应大小的空间
- 变量的存储类型
  - static
  - thread-local
  - extern
  - mutable

变量

说明变量

- 赋给该变量一标识符作为其名
- 指定其类型
- 在内存中分配给它一片存放空间，其大小由类型决定
- 使用说明语句
  - 在说明语句中，通过赋初值给它一个当前值，作为变量的初始值，该过程称为变量初始化

常变量与变量的存储



- 自动变量（C语言的概念）就是普通变量
- 外部变量存储与其所指变量的存储位置有关
- 还有一类存储于CPU寄存器中的变量，称为寄存器变量（register），也属于C语言的概念

数据与数据表示

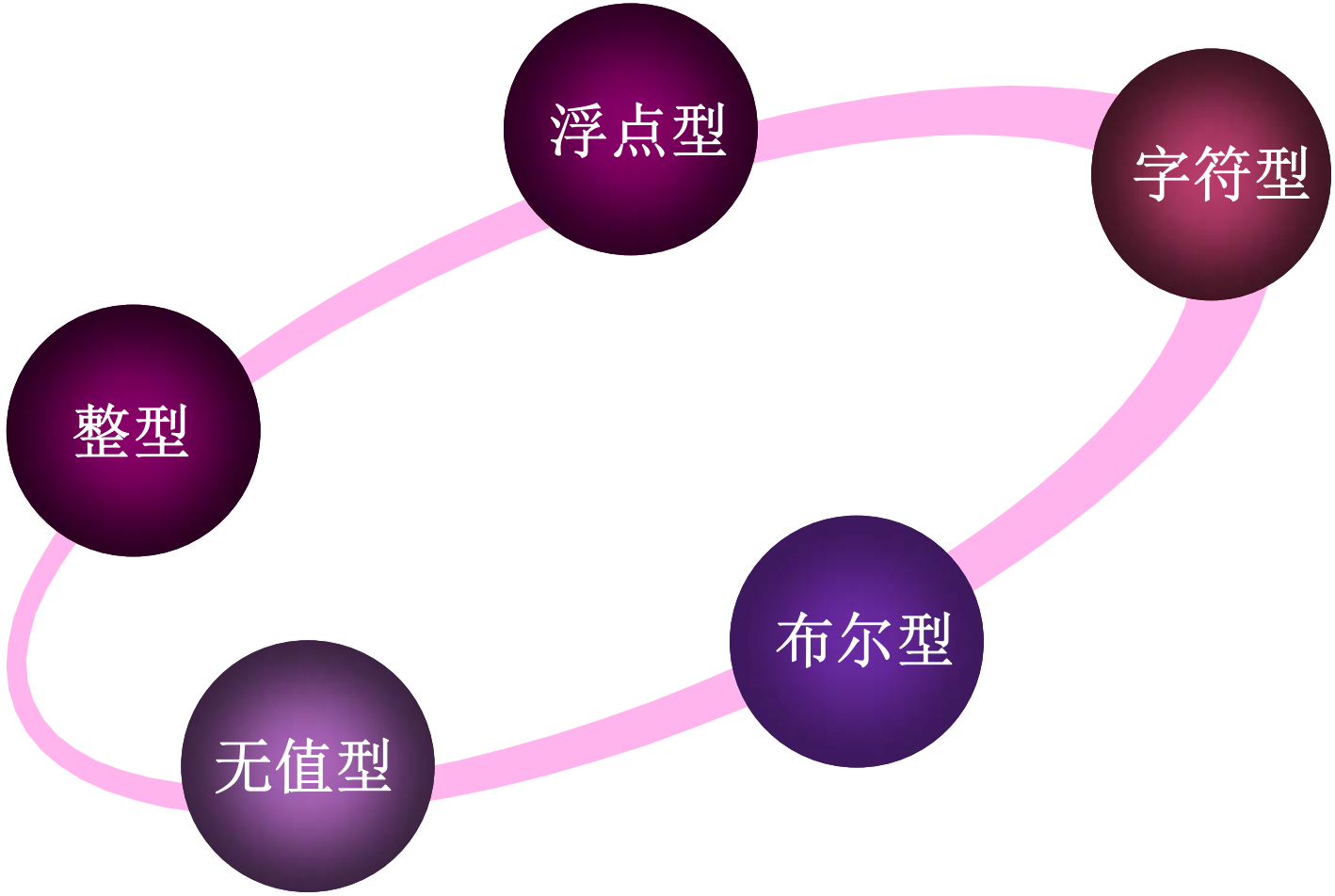
基本数据类型

复合数据类型

CV限定数据类型

说明语句

基本数据类型



整型

表示整数，用关键字int描述

- 正数、负数、0

最常用、最基本的数据类型

short、long、signed和unsigned描述不同类型的整型数据——整型的派生类型

C++语言规定整型数据占4个字节存储空间

- 短整型（short int）占2个字节
- 长整型（long int或long）占4个字节
- 长长整型（long long int 或long long）占8个字节

P59程序3.1

整型的修饰符 signed、unsigned

类型描述	字长	取值范围	简写
short int	2	-32768~32767	short
signed short int	2	-32768~32767(-2^{15}~2^{15}-1)	short
unsigned short int	2	0~65535(2^{16}-1)	unsigned short
int	4	-2^{31}~2^{31}-1	int
signed int	4	-2^{31}~2^{31}-1	int
unsigned int	4	0~4294967295(2^{32}-1)	unsigned
long int	4	-2147483648~2147483647	long
signed long int	4	-2147483648~2147483647	long
unsigned long int	4	0~4294967295	unsigned long
long long int	8	-2^{63}~2^{63}-1	long long
signed long long int	8	-2^{63}~2^{63}-1	long long
unsigned long long int	8	0~2^{64}-1	unsigned long long

浮点型

表示小数（或整个实数域）

- -1.2、0.0、2.5

分类

- 单精度浮点型
  - 占4个字节的存储空间，用关键字float描述
- 双精度浮点型
  - 占8个字节的存储空间，用关键字double描述
- 长双精度浮点型
  - 占10个字节的存储空间，用关键字long double描述

P60程序3.2

浮点型

浮点型及其派生类型

类型描述	字长	取值范围	精度
float	4	-3.4E38~3.4E38	1.0E-38
double	8	-1.7E308~1.7E308	1.0E-308
long double	10	-1.1E4932~1.1E4932	1.0E-4932

- 单精度浮点数的表示范围：
  - $1.00000000000000000000000 * 2^{-128} \approx 3.4e-38$
  - $1.11111111111111111111111 * 2^{127} \approx 3.4e38$
- 双精度浮点数的表示范围：
  - 计算方法类似于单精度

浮点型

浮点数无法表示全部的实数

- 浮点数最接近0的值

浮点数表示的实数通常是近似值

- 位数有限制，无法在有效的范围内表示

字符型

表示单个字符

- 基本字符96个
- 控制字符33个

ASCII字符占1个字节的存储空间

字符型的修饰符

- 用signed 和unsigned描述

类型描述	字长	取值范围	简写
char	1	-128~127	char
signed char	1	-128~127	char
unsigned char	1	0~255	unsigned char

ASCII字符型与整型的关系

字符型数据与在其表示范围内的整数等价

- 将字符型数据设置为整数

```
char myChar = 97;
```

- 将整型数据设置字符型

```
int myInt = 'a';
```

- 将字符型数据强制转换为整数

```
char myChar = 'a';
int b = static_cast<int>(myChar);
```

- 将整数强制转换为字符

```
int myInt = 97;
char b = (char)myInt;//C风格的数据类型强制转换
```

Unicode字符型

wchar_t

- 宽字符型
- 可以存储实现方式支持的最大扩展字符集中的所有成员
- 为字符型（char）字面值常量加前缀L
- 用wcin和wcout输入和输出

char16_t

- 为字符型（char）字面值常量加前缀u

char32_t

- 为字符型（char）字面值常量加前缀U

布尔型

用关键字bool描述，布尔型只包含两个值

- 真值：true，表示逻辑真
- 假值：false，表示逻辑假

含义

- 逻辑表达式或关系表达式的值
- 参加逻辑运算

布尔型

布尔型和整型的关系

- true与非0整数等价
  - 如果整型数据的值为-1，则其布尔值为true
  - 如果某个布尔型数据的值为true，则其整数值为1
- false与整数0等价

```
bool myBool = true;
int a = myBool;
cout<<a<<endl;
```

输出结果为1

```
int a = -5;
bool myBool = a;
cout<<myBool<<endl;
```

输出结果为1

无值型

用关键字void描述无值型

- 无值型的值域为空集

主要应用

- 声明函数的返回值
- 声明函数的参数
- 声明指针
  - void *<指针变量名>
    - 该指针可以指向任何数据类型

```
void main()

void main(void)
```

不能用void类型声明变量

数据与数据表示

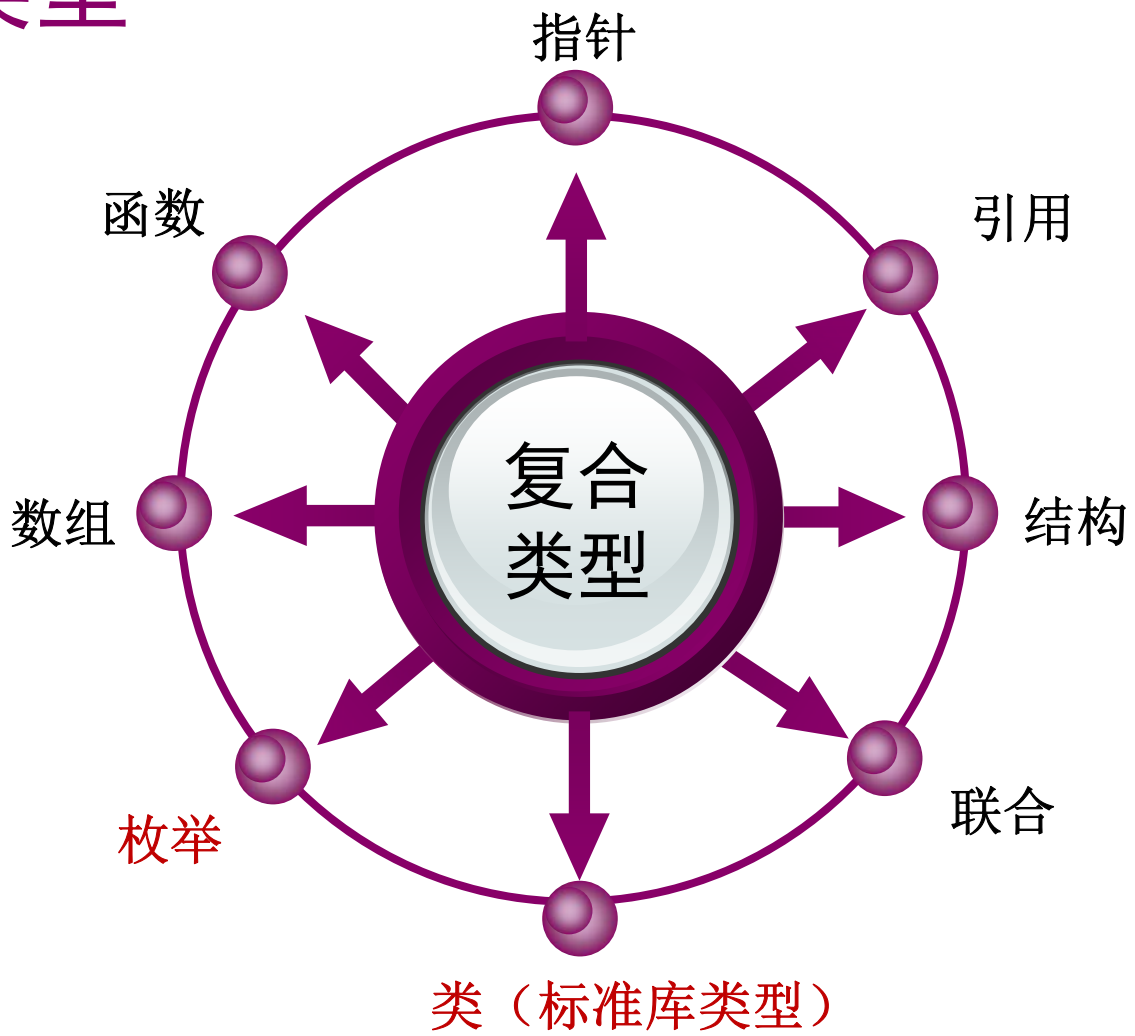
基本数据类型

复合数据类型

CV限定数据类型

说明语句

复合类型



枚举类型

用关键字enum描述枚举类型

将数据的取值（枚举值）列举出来，不能超出此范围取值

定义枚举类型

- enum<枚举类型名>{<枚举值表>}[<枚举变量表>]
  - 枚举类型名：标识符
  - 枚举值表：枚举类型数据的取值列表
    - 值名（是一个标识符）
    - <值名> = <整型常量>
  - 枚举变量表：声明为此类枚举类型的变量
    - enum <枚举类型名> <枚举变量表>

枚举类型

举例

- `enum color {RED=1,YELLOW,BLUE}c1=BLUE,c2;`
  - 枚举类型名：color
  - 枚举值
    - RED（对应整数1）
    - YELLOW（对应整数2）
    - BLUE（对应整数3）
  - 枚举类型变量
    - c1，初始值为BLUE
    - c2，未进行初始化

枚举类型

举例

- `color a, b = RED, c;`
  - 在color类型定义完毕后，可声明color类型的变量
    - 声明color变量时，关键字enum可以省略
  - 变量b的初始值为RED（或整数1）
- `enum day {Sun,Mon,Tue,Wed,Thu,Fri,Sat};`
  - Sun与整数0对应

```
enum day1 {Mon=1,Tue, Wed,Thu,Fri,Sat,Sun}
```

枚举类型

说明

- 枚举类型是整型的子集
- n个枚举值全部未赋整型常量值时，从左向右对应整数0~n-1
- 第i个枚举值赋予常量m，则从第i+1个枚举值开始，分别对应整数m+1，m+2，……，直到下一个赋了值的枚举值或结束
- 枚举类型变量不能赋整数值
- 枚举值按照常量处理，是整型常量
  - 枚举值用标识符表示，但是输出为整数

枚举类型

```
#include <iostream>
using namespace std;
void main() {
      enum color{RED,YELLOW=3,BLUE } c1=YELLOW,c2;
      color a, b=BLUE, c, d=RED;
      cout<<RED<<' '<<YELLOW<<' '<<BLUE<<endl;
      cout<<"c1="<<c1<<"  b="<<b<<"  d="<<d<<endl;
}
```

输出结果: 0 3 4
c1=3 b=4 d=0

字符串类型

字符串类

- 不是基本数据类型
- 在C++标准库中定义，属于标准库类型
- 可以作为数据类型使用

字符串对象

- 说明为字符串类型的变量
- 实际是字符串类的对象
- 字符串类型的某些功能通过字符串类的成员函数实现

字符串类型

说明字符串对象

```
string str;
```

字符串对象的初始化

```
string str{"hello"};
string str("hello");
string str="hello";
string str{};
```

字符串类型

操作字符串对象

- 访问字符串中的字符
  - 字符数组下标
  - 例如

```
string word ="Then";
word[2] = 'a';
```

  - 字符串对象的值变为：**"Than"**
  - 输入输出
  - **cin>>word;**
  - **cout<<word;**

字符串类型

操作字符串对象

- 计算长度
  - 调用字符串类的成员函数length()
  - 例如：`word.length()`
- 连接
  - 重载算术运算符“+”
  - 例如

```
string word1 = "hello";
string word2 = " C++";
string word3 = word1 + word2;
cout<<word3<<endl;//word3为："hello C++"
```

注意：+不支持两个字符串字面量的连接，如

```
string word4 = "hello" + " world!";
```

字符串类型

操作字符串对象

- 字符串连接字符、整数、浮点数

```
string str {"the result is "};
str += std::to_string('a');
str += std::to_string(69);
str += std::to_string(6.2832);
```

字符串类型

操作字符串对象

- 赋值

```
string word1="hello";
string word2{word1};//将word1值赋给word2
```

- 比较
  - 重载关系运算符：==、>、<、!=、>=、<=、
    - 逐个比较字符，直到找到第一个不同的字符，根据ASCII值比较不同字符的大小关系
  - 构造关于字符串的关系表达式

字符串类型

字符串数组

- 说明
  - `string str[10];`
- 初始化
  - `string str[10]={"zhao", "qian"};`
- 赋值
  - `str[2]="sun";`
- 存储方式
  - Visual C++编译器为字符串类型变量分配16Byte的存储空间，存储字符串在内存中的地址

字符串类型

宽字符串

- std::wstring

Unicode字符串

- std::u16string
- std::u32string

字符串类型

C样式的字符串

```
char *str;
char str[20];
```

将string字符串转换为C字符串

```
string string_type{"Hello"};
const char * c_type1 = string_type.c_str();
//常量指针
char *c_type2=string_type.data();//普通指针
```

数据类型的转换

隐式转换

- 运算表达式中，运算分量、运算结果的类型不一致时，支持部分数据类型的隐式转换
  - 赋值运算、算术运算等等

显式转换

- static_cast<目标数据类型>(表达式)
- const_cast<目标数据类型>(表达式)
- dynamic_cast<目标数据类型>(表达式)
- reinterpret_cast<目标数据类型>(表达式)

数据类型的转换

```
double d1=21.98,d2=6.28;
int i {static_cast<int>(d1+d2)};
int j {static_cast<int>(d1)+static_cast<int>(d2)};
```

程序执行后，变量i和j的值分别是：？

数据与数据表示

基本数据类型

复合数据类型

CV限定数据类型

说明语句

CV限定符（CV-qualifiers）

const

- 常量限定，用于将某个变量限定为其值不可变（常变量）

volatile

- **volatile**提醒编译器它后面所定义的变量随时都有可能改变，因此编译后的程序每次需要存储或读取这个变量的时候，都会直接从变量地址中读取数据。
- 如果没有volatile关键字，则编译器可能优化读取和存储，可能暂时使用寄存器中的值，如果这个变量由别的程序更新了的话，将出现不一致的现象。

数据与数据表示

基本数据类型

复合数据类型

CV限定数据类型

说明语句

C++语句概览

- 标签语句
- 表达式语句
  - 数学表达式
  - 赋值语句
  - 输入输出语句
  - 函数调用
  - ……
- 分支语句
- 循环语句
- 转向语句
- 复合语句和语句块
- 说明语句

常量的说明

常变量说明及初始化

- const <数据类型> <常变量名> = <表达式>
  - 类型名，即数据类型
  - 常变量名是标识符
  - 表达式的类型与常变量类型一致

宏替换定义常量

- 字符串替换，没有数据类型，不分配存储空间
- #define <常量名> <表达式>

变量的说明

变量说明

- [<存储类属性>] <数据类型> <变量名>[,<变量名>]*;

变量的存储类属性

- static：静态变量
- thread_local：线程的局部变量
- extern：外部变量
- mutable：可变变量
- auto
- register

有关变量的概念

全局变量和局部变量

- 全局变量：程序运行的整个过程，内存中位置不变
- 局部变量：程序运行的过程中，所占内存反复占用和释放

生存期和作用域

- 生存期：变量所占内存空间由分配到释放的时期
- 作用域：变量有效的范围，说明变量时所在的语句

初始化（Initialization）

变量的初始化

- 在变量说明语句中对变量进行赋值
- 格式
  - [<存储类属性>] <数据类型> <变量名>{<初始值>};
  - [<存储类属性>] <数据类型> <变量名>(<初始值>);
  - [<存储类属性>] <数据类型> <变量名>=<初始值>;

```
int a{5}; //初始化列表法，默认初始值为0，如int a{};
int a(5); //函数表示法
int a = 5;  //赋值表示法
```

定义（definition）

带有初始化的说明语句

[存储类属性] <数据类型> <变量名表>;

变量名表

- 可以完成对变量的初始化
- 格式

<变量名>[= <表达式>][, <变量名表>]*;

举例

```
float a, b;
float a = 3.14, b;
```

命名空间的说明

C++语言的新标准引入的概念

由用户命名的作用域，解决大型程序中标识符重名的问题

- 说明语句格式

  namespace <标识符>{<若干说明或定义>}

- 使用方式

  using namespace <名字>;

auto关键字

告诉编译器应该“推断”类型

- 例如

```
auto m {10}; //m被推断为整型
auto n = 200UL; //n被推断为无符号长整型
auto pai(3.1416);  //pai被推断为双精度浮点型
```

using 关键字

为数据类型设置别名

- 例如：

```
using BigOnes = unsigned long long;
//BigOnes可以用来代替无符号长长整型
BigOnes a = 100;
```

typedef关键字

类型说明

- 用户为一个已定义的类型赋予一个新的数据类型名
- 用来提高程序的可读性

```
typedef int id_number;
```

  - 可以用**id_number**表示整型

```
id_number student_id = 1901234;
```

第二章 结束