Technology Tarena 达内科技

标准C++语言

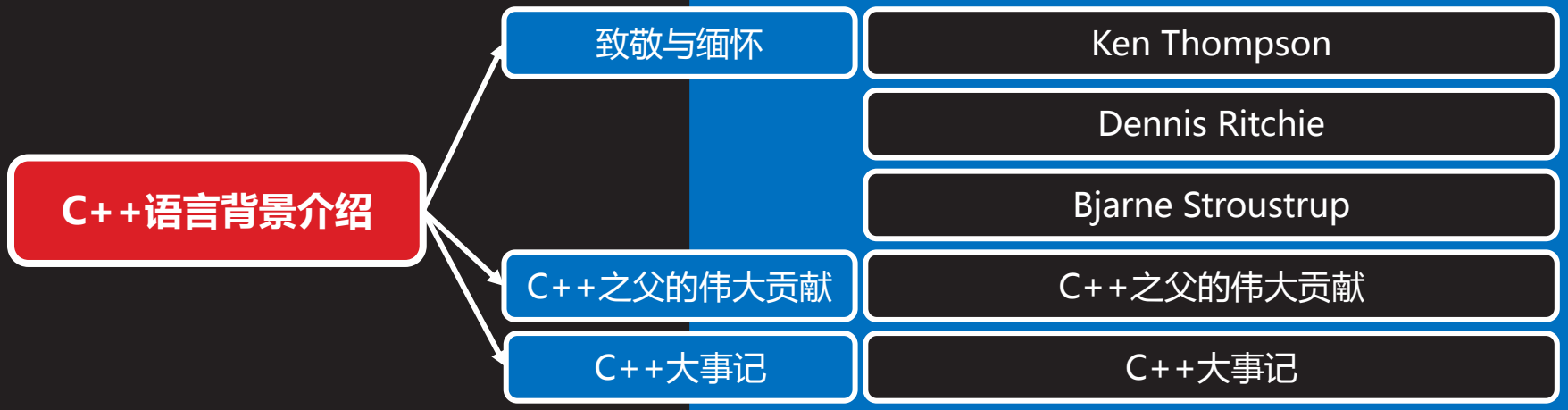
PART 1

DAY01

内容

上午	09:00 ~ 09:30	C++语言背景介绍
	09:30 ~ 10:20	
	10:30 ~ 11:20	第一个C++程序
	11:30 ~ 12:20	名字空间
下午	14:00 ~ 14:50	结构、联合和枚举
	15:00 ~ 15:50	字符串
	16:00 ~ 16:50	布尔型与操作符别名
	17:00 ~ 17:30	总结和答疑

C++语言背景介绍


C++语言背景介绍
致敬与缅怀
Ken Thompson
Dennis Ritchie
Bjarne Stroustrup
C++之父的伟大贡献
C++之父的伟大贡献
C++大事记
C++大事记

Technology Tarena 达内科技

致敬与缅怀

Ken Thompson

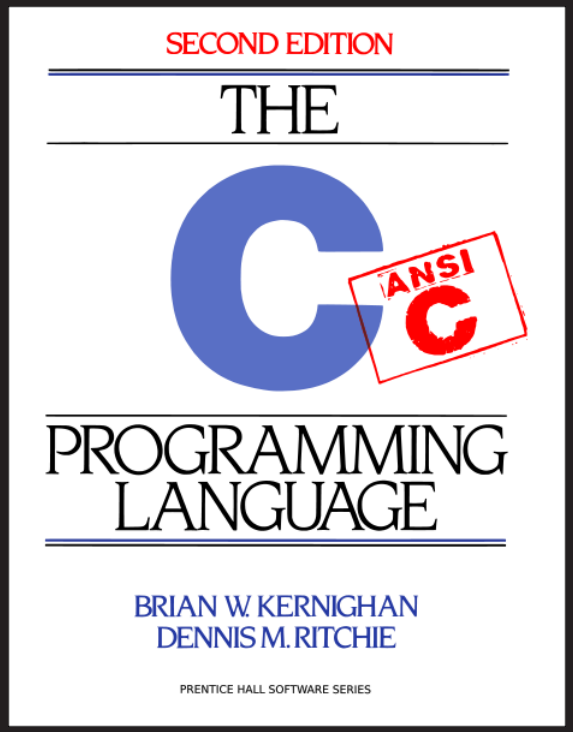
- Ken Thompson (肯•汤普逊)，1943-，B语言之父、UNIX发明人之一

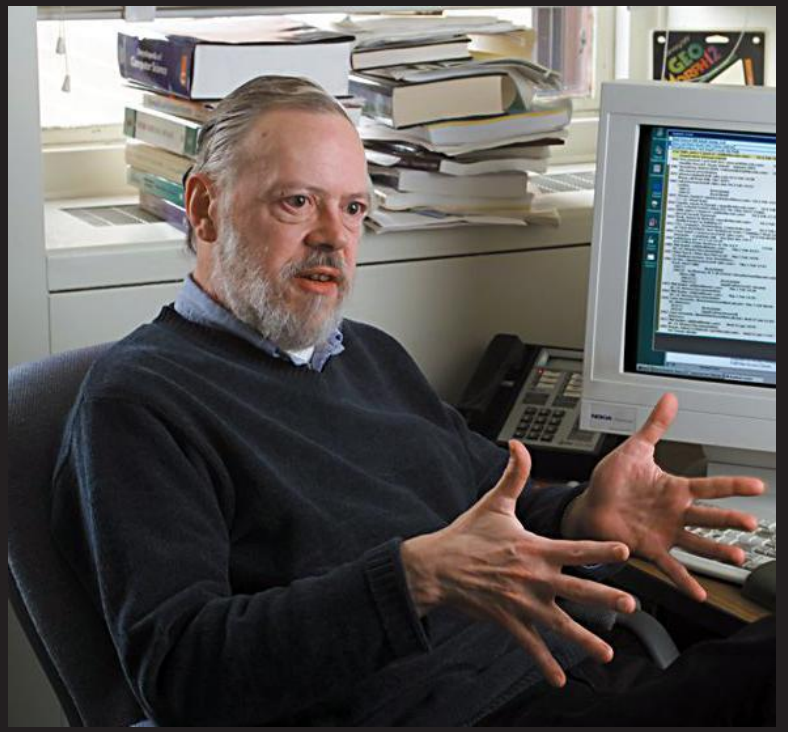
知识讲解

Dennis Ritchie

- Dennis M. Ritchie (丹尼斯·里奇)，1941-2011，C语言之父、UNIX之父、黑客之父

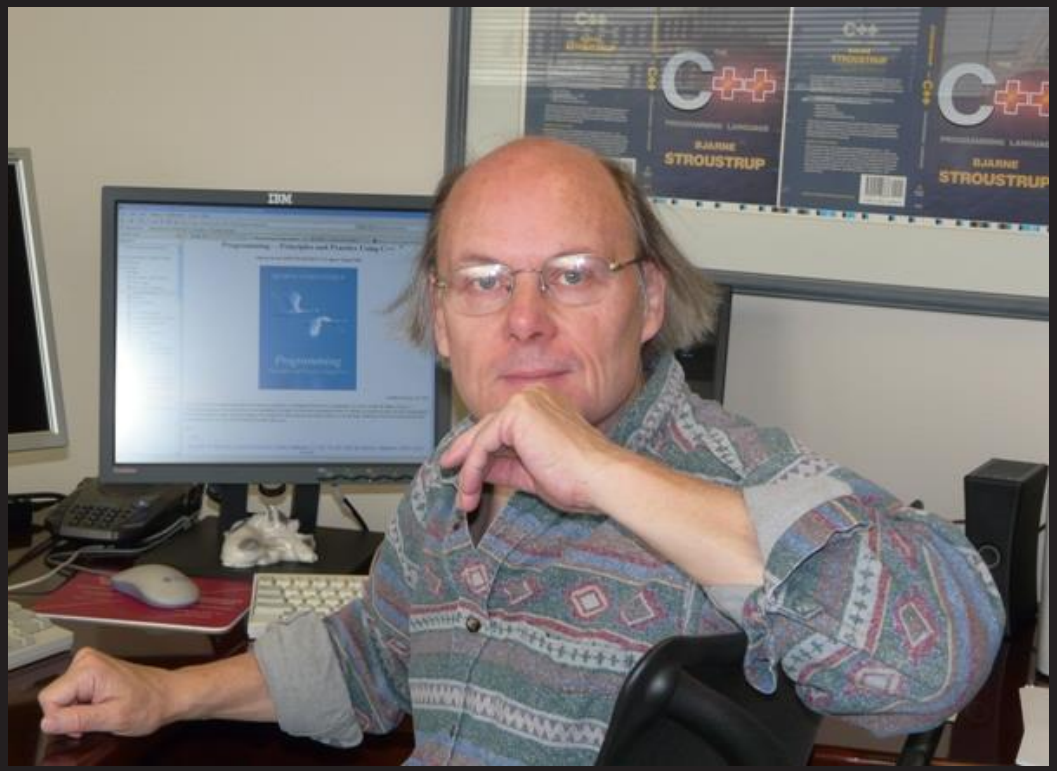
知识讲解





Bjarne Stroustrup

- Bjarne Stroustrup (本贾尼·斯特劳斯特卢普)，1950-，C++之父



知识讲解

Technology Tarena 达内科技

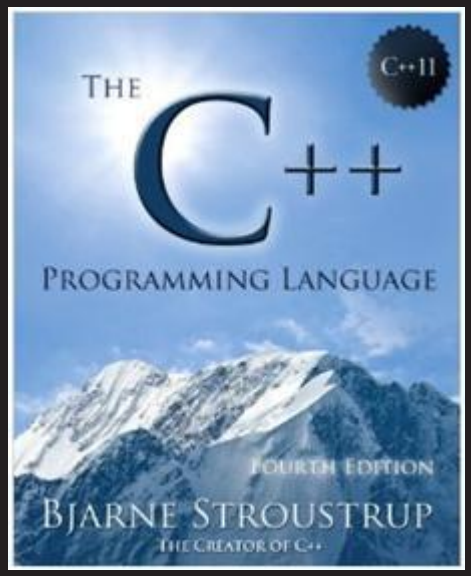
C++之父的伟大贡献

C++之父的伟大贡献

- UNIX系统分布内核流量分析
  - 1979年4月，Bjarne博士试图寻找一种有效的工具，以更加模块化的方法，分析UNIX系统由于内核分布而造成的网络流量
- Cpre
  - 1979年10月，Bjarne博士完成了一个预处理程序—Cpre，为C增加了类似Simula的类机制
- C with Classes
  - 1983年，Bjarne博士开发了一种全新的编程语言，C with Classes，即后来的C++
  - C++在运行时间、代码紧凑性和数据紧凑性方面，完全可以和C语言相媲美

C++之父的伟大贡献（续1）

- 不断汲取其它语言的精华
  - Simula的类
  - Algol68的运算符重载和引用
  - BCPL的“//”注释
  - Ada的模板和名字空间
  - Clu和ML的异常
- CFront 1.0
  - 1985年，CFront 1.0发布
  - Bjarne博士推出经典巨著《The C++ Programming Language》第一版



知识讲解

Technology Tarena 达内科技

C++大事记

C++大事记

- 1979：Bjarne博士在贝尔实验室完成了Cpre
- 1983：第一个C with Classes实现投入使用
- 1985：CFront 1.0发布
- 1987：GNU C++发布
- 1990：Borland C++发布
- 1992：Microsoft C++发布，IBM C++发布
- 1998：ISO标准被批准，即C++98
- 2003：ISO对C++98进行修订，即C++03
- 2011：ISO发布ISO/IEC 15882:2011，即C++11
- 2014：ISO对C++11做了部分扩展，即C++14

第一个C++程序

Technology Tarena 达内科技

Hello, World !

似曾相识

- 编译器
  - g++
  - 也可以用gcc，但要加上-lstdc++
- 扩展名
  - .cpp/.cc/.C/.cxx
  - 也可以用.c，但要加上-x c++
- 头文件
  - #include <iostream>
  - 也可以#include <cstdio>

知识讲解

似曾相识（续1）

- I/O流
  - cout <<
  - cin >>
  - 也可以用scanf/printf
- 名字空间
  - std::
  - using namespace std;
  - 所有标准类型、对象和函数都位于std命名空间中

知识讲解

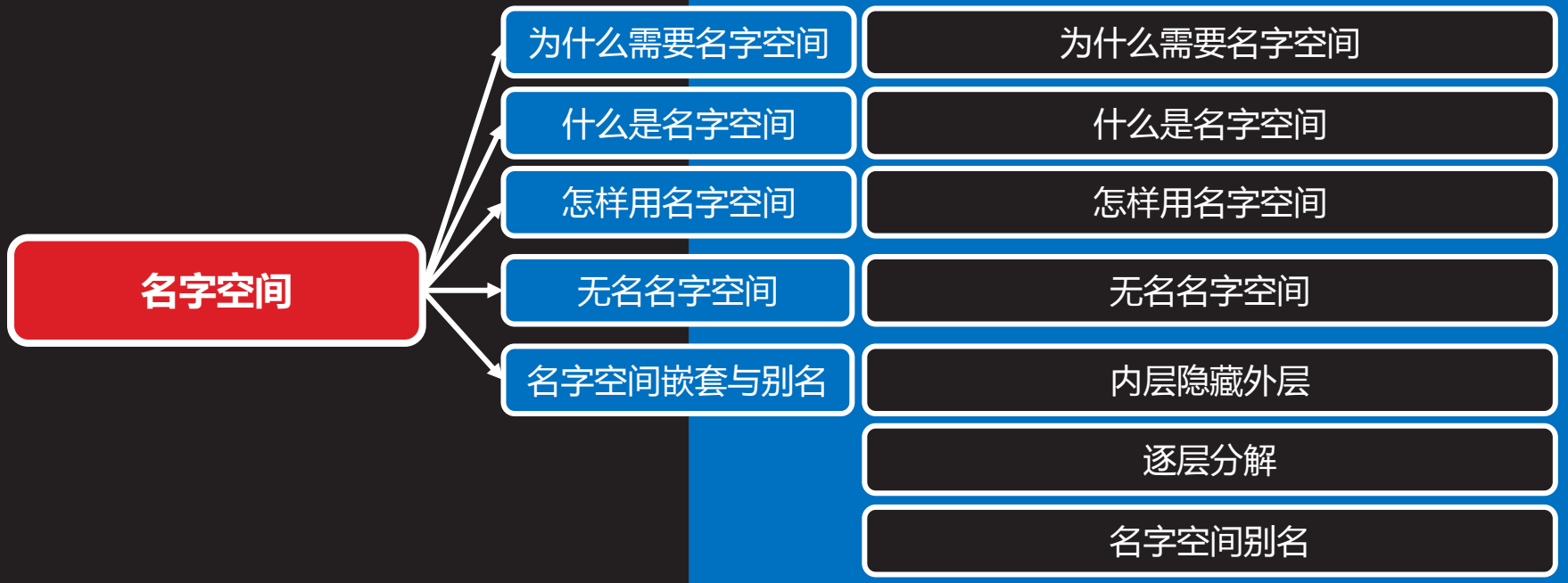
Hello, World !

【参见：TTS COOKBOOK 】

课堂练习

- Hello, World !

名字空间


名字空间
为什么需要名字空间
什么是名字空间
怎样用名字空间
无名名字空间
名字空间嵌套与别名
为什么需要名字空间
什么是名字空间
怎样用名字空间
无名名字空间
内层隐藏外层
逐层分解
名字空间别名

Technology Tarena 达内科技

为什么需要名字空间

为什么需要名字空间

- 划分逻辑单元
  - 文件是物理单元，而名字空间则是逻辑单元
- 避免名字冲突
  - 同一个作用域中不允许同时存在名字相同的标识符

知识讲解

Technology Tarena 达内科技

什么是名字空间

什么是名字空间

- 名字空间定义
  - namespace 名字空间名 {
    名字空间成员1;
    名字空间成员2; }
- 名字空间合并
  - namespace 名字空间名 {
    名字空间成员3; }
- 声明定义分开
  - 名字空间的声明和定义可以分开，但是定义部分需要借助作用域限定操作符“::”指明所定义的成员隶属于哪个名字空间

知识讲解

名字空间的定义

【参见：TTS COOKBOOK 】

课堂练习

- 名字空间的定义

Technology Tarena 达内科技

怎样用名字空间

怎样用名字空间

- 作用域限定符
  - 名字空间名::名字空间成员名
- 名字空间指令
  - using namespace 名字空间名;
  - 名字空间指令以后的代码，对名字空间中的所有成员可见，对可见名字空间成员的引用，可省略作用域限定符
  - 名字空间指令有可能导致名字冲突
- 名字空间声明
  - using 名字空间名::名字空间成员名;
  - 名字空间声明用于将名字空间中的特定标识符引入当前作用域，可省略作用域限定符，直接引用之
  - 名字空间声明有可能导致重复声明

知识讲解

名字空间的使用

【参见：TTS COOKBOOK 】

课堂练习

- 名字空间的使用

Technology Tarena 达内科技

无名名字空间

无名名字空间

- 不属于任何有名名字空间的标识符，隶属于无名命名空间
- 无名命名空间的成员，直接通过“::”访问，特别是在其与有名名字空间中的同名标识符构成冲突或被后者隐藏时
  - int var = 100;
    namespace ns {
        int var = 200;
        void fun (void) {
            cout << var << ' ' << ::var << endl; // 200 100 } }
  - using namespace ns;
    cout << var << endl; // 错误
    cout << ns::var << ' ' << ::var << endl; // 200 100
  - using ns::var;
    cout << var << ' ' << ::var << endl; // 200 100

知识讲解

无名名字空间

【参见：TTS COOKBOOK 】

课堂练习

- 无名名字空间

Technology Tarena 达内科技

名字空间嵌套与别名

内层隐藏外层

- 名字空间内部可以再定义名字空间，谓之名字空间嵌套
  - namespace ns1 { namespace ns2 { … } }
- 在内层名字空间声明的标识符，隐藏在外层名字空间声明的同名标识符
  - namespace ns1 {

```
        int var = 1;
        namespace ns2 {
            int var = 2;
            void fun (void) {
                cout << var << endl; // 2
            }
        }
    }
```

知识讲解

逐层分解

- 无论是定义还是使用嵌套的名字空间中的标识符，都需要通过作用域限定操作符逐层分解
  - namespace ns1 {
    namespace ns2 {
    namespace ns3 {
    extern int var;
    void fun (void); } } }
  - int ns1::ns2::ns3::var = 123;
    void ns1::ns2::ns3::fun (void) { cout << var << endl; }
  - cout << ns1::ns2::ns3::var << endl; // 123
    ns1::ns2::ns3::fun (); // 123

名字空间别名

- 利用名字空间别名，可以简化名字空间路径的书写形式
  - namespace ns1 {
    namespace ns2 {
    namespace ns3 {
    extern int var;
    void fun (void); } } }
  - namespace ns123 = ns1::ns2::ns3;
  - int ns123::var = 123;
    void ns123::fun (void) { cout << var << endl; }
  - cout << ns123::var << endl; // 123
    ns123::fun (); // 123

知识讲解

多层名字空间

【参见：TTS COOKBOOK 】

课堂练习

- 多层名字空间

结构、联合和枚举

Technology Tarena 达内科技

C++的结构

C++的结构

- 声明或定义结构型变量，可以省略struct关键字
- 可以定义成员函数，在结构体的成员函数中可以直接访问该结构体的成员变量，无需通过“.”或“->”
  - struct User {
    char name[256];
    int age;
    void who (void) {
    cout << "我是" << name <<
    "，今年" << age << "岁。" << endl; }
    };
  - User user = {"张飞", 25}, *puser = &user;
    user.who ();
    puser->who ();

知识讲解

Technology Tarena 达内科技

C++的结构

【参见：TTS COOKBOOK 】

课堂练习

- C++的结构

Technology Tarena 达内科技

C++的联合

C++的联合

- 声明或定义联合型变量，可以省略union关键字
- 与其说匿名联合是一种类型定义，倒不如说是一种对多个变量在内存中布局方式的表达
  - union {
    int n;
    char c[sizeof (n)];
    };
  - n = 0x12345678;
  - for (size_t i = 0; i < sizeof (c); ++i)
    printf ("%#x ", c[i]); // 0x78 0x56 0x34 0x12
    printf ("\n");

知识讲解

C++的匿名联合

【参见：TTS COOKBOOK 】

课堂练习

- C++的匿名联合

Technology Tarena 达内科技

C++的枚举

C++的枚举

- 声明或定义枚举型变量，可以省略enum关键字
- C++中的枚举是一种独立的数据类型，整型不能隐式转换为枚举型
  - enum Color {
    RED, YELLOW, BLUE, WHITE, BLACK } color;
  - color = RED;
    color = 5; // 错误
    color = 0; // 错误
  - void foo (Color color);
    foo (RED);
    foo (5); // 错误
    foo (0); // 错误

知识讲解

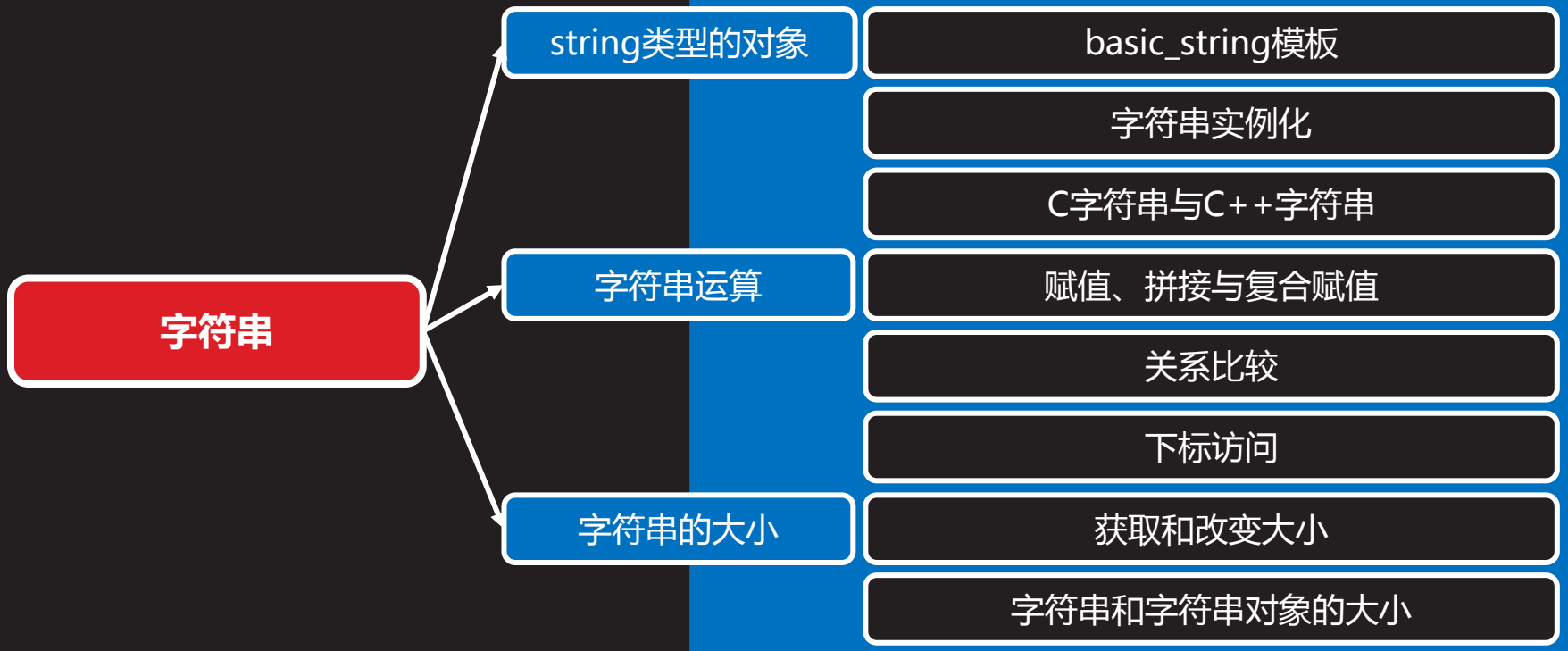
C++的枚举

【参见：TTS COOKBOOK 】

课堂练习

- C++的枚举

字符串


字符串
string类型的对象
basic_string模板
字符串实例化
C字符串与C++字符串
字符串运算
赋值、拼接与复合赋值
关系比较
下标访问
字符串的大小
获取和改变大小
字符串和字符串对象的大小

string类型的对象

basic_string模板

- C++中的string实际上是basic_string模板被char类型参数实例化后的类型别名
  - template<typename charT, ...> class basic_string { ... };
  - typedef basic_string<char> string;
- 需要包含专门的头文件<string>
  - 注意区分以下三个头文件
    - ✓ `<string>` // C++的string
    - ✓ `<cstring>` // C的字符串，如strlen、strcpy、strcat等
    - ✓ `<string.h>` // 等价于<cstring>

知识讲解

字符串实例化

- 在栈中隐式实例化
  - string str;
  - string str ("Hello, World !");
- 在栈中显式实例化
  - string str = string ();
  - string str = string ("Hello, World !");
- 在堆中实例化
  - string* str = new string;
  - string* str = new string ();
  - string* str = new string ("Hello, World !");

知识讲解

C字符串与C++字符串

- C++的string支持源自C风格字符指针的隐式类型转换
  - string str = "Hello, World !";
  - char const* pc = "Hello, World !";
    string str = pc;
  - char sa[] = "Hello, World !";
    string str = sa;
- C++的string提供了获取C风格字符指针的成员函数
  - char const* pc = str.c_str ();
  - c_str()返回字符串对象中用于存放其字符内容的内存地址

知识讲解

Technology Tarena 达内科技

字符串运算

赋值、拼接与复合赋值

- 赋值
  - string s1 = "hello", s2 = "world";
    s1 = s2;
    cout << s1 << endl; // world
- 拼接
  - string s1 = "hello", s2 = "world";
    string s3 = s1 + s2;
    cout << s3 << endl; // helloworld
- 复合赋值
  - string s1 = "hello", s2 = "world";
    s1 += s2;
    cout << s1 << endl; // helloworld

知识讲解

关系比较

- </<=/==/!=/>=/>
  - string s1 = "zhangfei";
    string s2 = "zhaoyun";
    if (s1 == s2)
        cout << s1 << "==" << s2 << endl;
    else if (s1 < s2)
        cout << s1 << "<" << s2 << endl;
    else
        cout << s1 << ">" << s2 << endl;
- string类型的关系比较区分大小写
  - string ("zhangfei") < string ("zhaoyun")
  - string ("zhangfei") > string ("zhaOyun")

知识讲解

下标访问

- 通过下标访问字符串中的单个字符
  - string str = "hello, world !";
    for (size_t i = 0; i < str.length (); ++i)
        if ('a' <= str[i] && str[i] <= 'z')
            str[i] -= 'a' - 'A';
    cout << str << endl; // HELLO, WORLD !
- 不检查下标越界
  - string str = "hello, world !";
    str[14] = 'A'; // 未定义

知识讲解

Technology Tarena 达内科技

字符串的大小

获取和改变大小

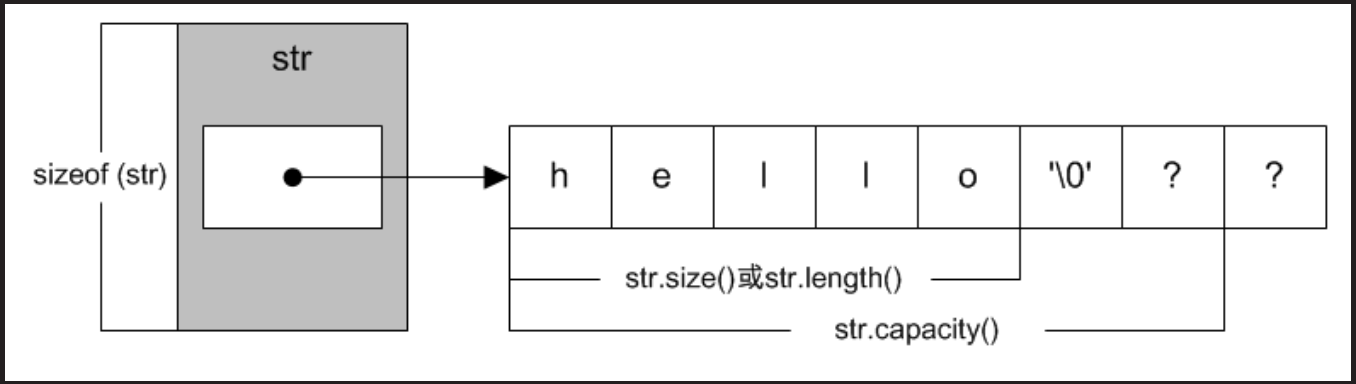
- 获取大小，即字符数
  - size_type size (void) const;
  - size_type length (void) const;
  - 不包括结尾空字符
- 改变大小，可增可减
  - void resize (size_type size, const char& ch = '\0');
  - 新增部分用第二个参数所表示的字符初始化，缺省空字符
- 清空字符串内容
  - void clear (void);
- 判断是否为空，空返回true，不空返回false
  - bool empty (void) const;

知识讲解

字符串和字符串对象的大小

- 字符串大小指其实际容纳的字符数，不包括结尾空字符
- 字符串对象大小指该对象本身的字节数，不包括其内容
- 多数C++实现，string实际上只包含一个char*类型的非静态成员变量，该变量保存一个以空字符结尾的字符数组首地址。因此，一个string对象的大小，实际上就是它唯一一个非静态成员变量的大小，4个字节(32位系统)

知识讲解



C++字符串的基本用法

【参见：TTS COOKBOOK 】

课堂练习

- C++字符串的基本用法

布尔型与操作符别名

Technology Tarena 达内科技

布尔类型

布尔类型

- 表示布尔量的数据类型
  - bool
- 布尔类型的字面值常量
  - true表示真
  - false表示假
- 以单字节整数形式存储，类似于char
  - 分别用1和0表示真和假
- 任何基本类型都可以被隐式转换为布尔类型
  - 非0即真，0即假

知识讲解

布尔类型

【参见：TTS COOKBOOK 】

课堂练习

- 布尔类型

Technology Tarena 达内科技

操作符别名

操作符别名

- 某些欧洲语言所使用的字母比26个基本拉丁字母多，占用了键盘中“~”、“&”等特殊符号的位置
- 国际标准化组织为一些操作符规定了别名，以便使用这些语言的键盘也能输入正确的C/C++代码
- C95和C++98以后的语言标准都支持ISO-646

知识讲解

主名	别名	主名	别名	主名	别名
{	<%	&&	and	|	bitor
}	%>	||	or	|=	or_eq
[	<:	!	not	^	xor
]	:>	!=	not_eq	^=	xor_eq
#	%:	&	bitand	~	compl
##	%:%:	&=	and_eq		

操作符别名（续1）

知识讲解

```
#include <iostream>
using namespace std;
int main (void) <%
    int a<::> = <%13, 21, 37, 49, 58%>;
    int b<::> = <%79, 63, 45, 31, 16%>;
    for (int i = 0; i < 5; ++i)
        cout << (a<:i:> bitand b<:i:>) +
            ((a<:i:> xor b<:i:>) >> 1) << ' ';
    cout << endl;
%>
```

Technology Tarena 达内科技

总结和答疑