Technology Tarena 达内科技

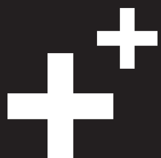
标准C++语言

PART 2

DAY04

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	虚析构
	10:30 ~ 11:20	异常
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	I/O流
	15:00 ~ 15:50	
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑



虚析构

虚析构

虚析构函数

- 子类对象的内存泄漏
- 虚析构函数
- 空虚析构函数

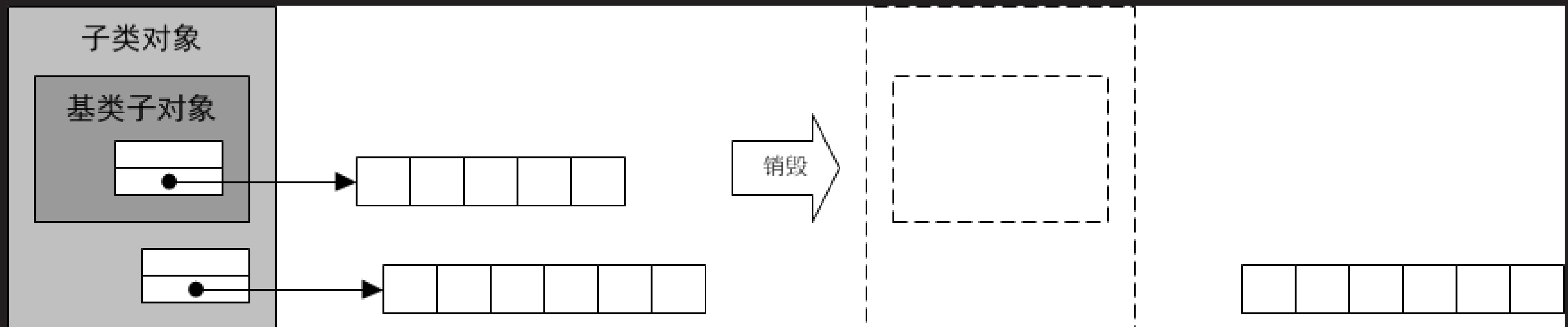
Technology
Tarena
达内科技

子类对象的内存泄漏

Tarena Technology 达内科技

- delete一个指向子类对象的基类指针
  - 实际被调用的仅仅是基类的析构函数
  - 基类的析构函数负责析构子类对象中的基类子对象
  - 基类的析构函数不会调用子类的析构函数
  - 在子类中分配的资源将形成内存泄漏

知识讲解

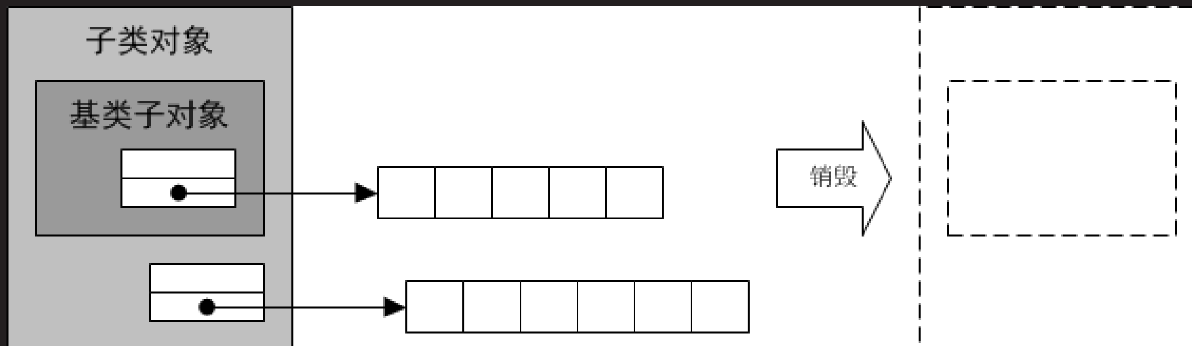


虚析构函数

Tarena Technology 达内科技

- 如果将基类的析构函数声明为虚函数，那么当delete一个指向子类对象的基类指针时，实际被调用的将是子类的析构函数
- 子类的析构函数将首先析构子类对象的扩展部分，然后再通过基类的析构函数析构该对象的基类部分，最终实现完美的资源释放

知识讲解



Tarena Technology 达内科技

空虚析构函数

知识讲解

- 没有分配任何资源的类，无需定义析构函数
- 没有定义析构函数的类，编译器会为其提供一个缺省析构函数，但缺省析构函数并不是虚函数
- 为了保证delete一个指向子类对象的基类指针时，能够正确调用子类的析构函数，就必须把基类的析构函数定义为虚函数，即使它是一个空函数
- 任何时候，为基类定义一个虚析构函数总是无害的

Technology Tarena 达内科技

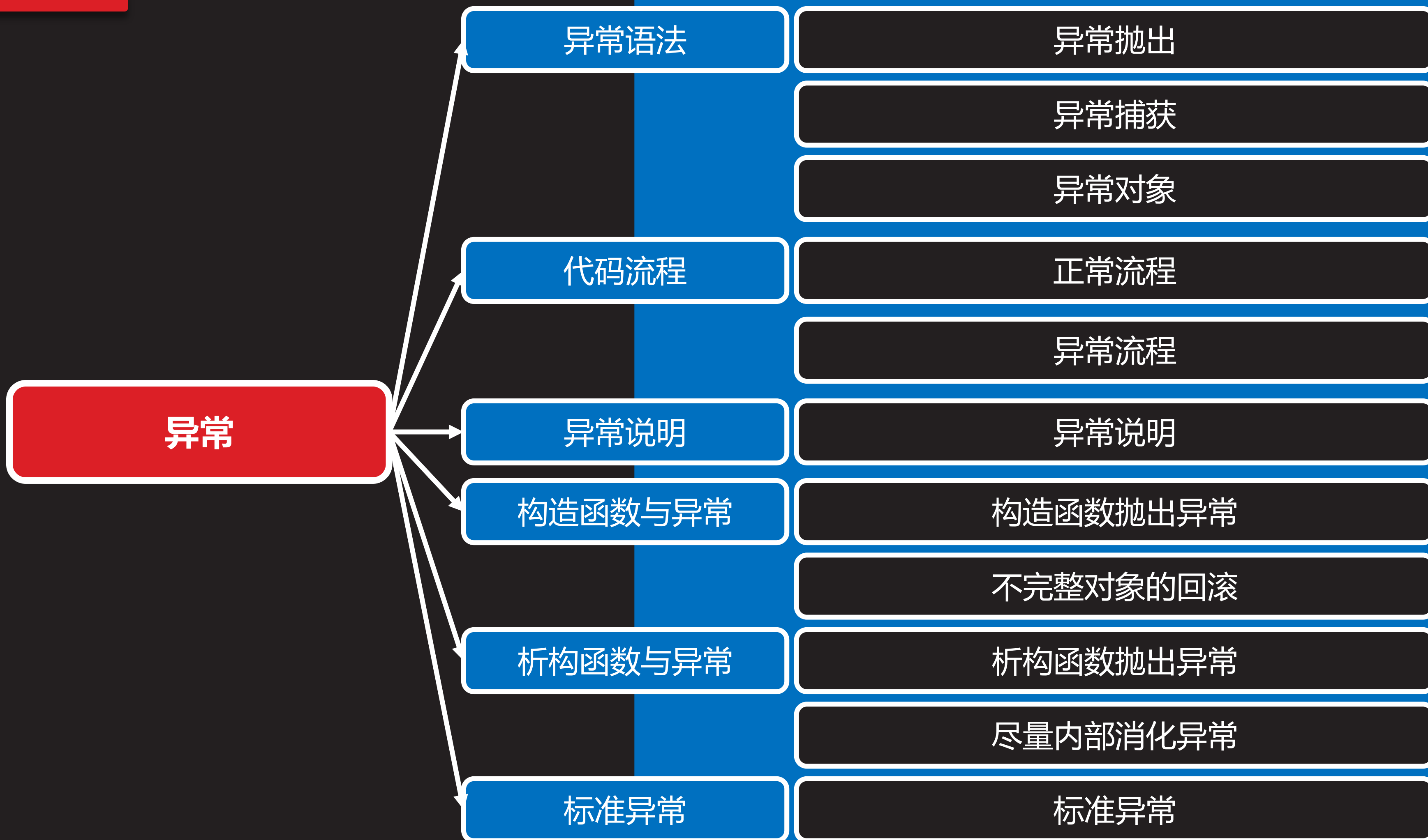
虚析构函数

【参见：TTS COOKBOOK 】

课堂练习

- 虚析构函数

异常



Technology Tarena 达内科技

异常抛出

知识讲解

- throw 异常对象;
- 可以抛出基本类型的对象
  - throw -1;
  - throw "打开文件失败！";
- 可以抛出类类型的对象
  - FileException ex ("打开文件失败！");
    throw ex;
  - throw FileException ("打开文件失败！");
- 不要抛出局部对象的指针
  - FileException ex ("打开文件失败！");
    throw &ex; // 错误！

异常捕获

Tarena Technology 达内科技

知识讲解

- try {
    可能引发异常的语句;
  }
  catch (异常类型1& ex) {
    针对异常类型1的异常处理;
  }
  catch (异常类型2& ex) {
    针对异常类型2的异常处理;
  }
  catch (...) {
    针对其它异常类型的异常处理;
  }

异常捕获（续1）

Tarena Technology 达内科技

- 根据异常对象的类型自上至下顺序匹配，而非最优匹配，因此对子类类型异常的捕获不要放在对基类类型异常的捕获后面
  - class GeneralException { ... };
    class FileException : public GeneralException { ... };
  - try { ... }
    catch (FileException& ex) { ... }
    catch (GeneralException& ex) { ... }
- 建议在catch子句中使用引用接收异常对象，避免因为拷贝构造带来性能损失，或引发新的异常
  - try { ... }
    catch (FileException& ex) { ... }

知识讲解

异常对象

Tarena Technology 达内科技

知识讲解

- 为每一种异常情况定义一种异常类型
  - class FileException { ... };
  - class MemoryException { ... };
  - class SocketException { ... };
- 推荐以匿名临时对象的形式抛出异常
  - throw FileException (...);
- 异常对象必须允许被拷贝构造和析构
- 建议从标准库异常中派生自己的异常
  - #include <stdexcept>
  - class FileException : public exception { ... };

异常对象（续1）

Tarena Technology 达内科技

知识讲解

- 当执行一个throw语句时，运行期异常处理机制会把异常对象复制到一个临时对象中，该临时对象存在于某个“安全区”中
- 存储临时异常对象的“安全区”高度平台相关，但可以保证在最后一个使用该异常对象的catch块完成之前，该对象都一直保持可用状态
- 当异常发生时，没有什么地方是绝对安全的，除了这个存放临时异常对象的“安全区”，这里是唯一风平浪静的风暴之眼
- 由此可见，将一个对象的指针作为异常抛出是不明智的，而通过引用访问“安全区”中的异常对象则是安全的

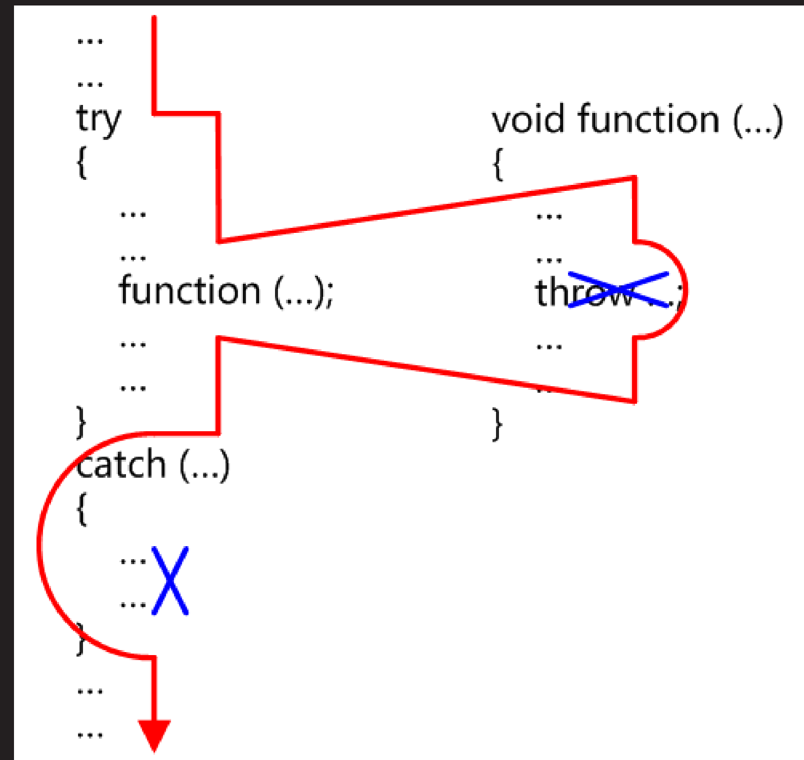
Technology Tarena 达内科技

正常流程

Tarena Technology 达内科技

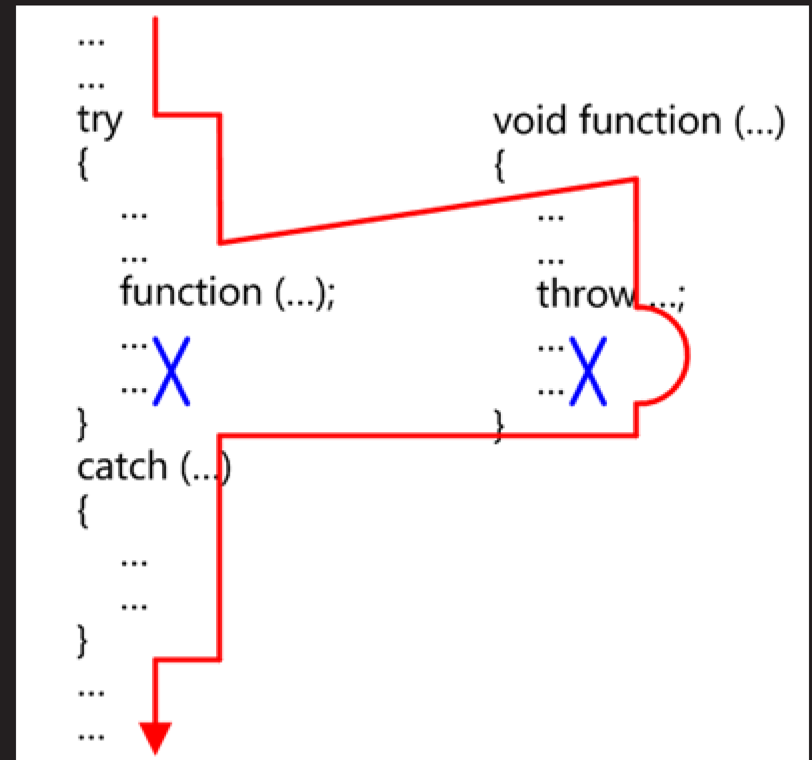
- 两个执行
  - 函数中throw语句之后的代码执行
  - try块中函数调用语句之后的代码执行
- 两个不执行
  - throw语句不执行
  - catch块不执行

知识讲解



异常流程

- 两个不执行
  - 函数中throw语句之后的代码不执行
  - try块中函数调用语句之后的代码不执行
- 两个执行
  - throw语句执行
  - catch块执行



知识讲解

Tarena Technology 达内科技

异常说明

知识讲解

- 异常说明是函数原型的一部分，旨在说明函数可能抛出的异常类型
  - 返回类型 函数名 (形参表)
    throw (异常类型1, 异常类型2, ...) {
    函数体; }
  - void connectServer (char const* config)
    throw (int, string, FileException) {
    ... }
- 异常说明是一种承诺，承诺函数不会抛出异常说明以外的异常类型
  - 如果函数抛出了异常说明以外的异常类型，那么该异常将无法被捕获，并导致进程中止

异常说明（续1）

- 隐式抛出异常的函数也可以列出它的异常说明
  - void connectServer (char const* config)
    throw (int, string, FileException);
  - void login (char const* username, char const* passwd)
    throw (int, string, FileExcetion) {
    connectServer ("/etc/server.cfg"); }
- 没有异常说明，表示可能抛出任何类型的异常
  - void connectServer (char const* config);
- 异常说明为空，表示不会抛出任何类型的异常
  - void connectServer (char const* config) throw ();

异常说明（续2）

Tarena Technology 达内科技

- 如果基类中的虚函数带有异常说明，那么该函数在子类中的覆盖版本不能说明比其基类版本抛出更多的异常
  - class A {
    virtual void foo (void) throw (int, string);
    virtual void bar (void) throw (); };
  - class B : public A {
    void foo (void) throw (double); // 错误
    void bar (void) throw (); };
- 异常说明在函数的声明和定义中必须保持严格一致，否则将导致编译错误

知识讲解

异常说明

【参见：TTS COOKBOOK 】

- 异常说明

Technology
Tarena
达内科技

构造函数抛出异常

Tarena Technology 达内科技

- 构造函数可以抛出异常，某些时候还必须抛出异常
  - 构造过程中可能遇到各种错误，比如内存分配失败
  - 构造函数没有返回值，无法通过返回值通知调用者
  - class String {

```
class String {
public:
    String (char const* str) {
        if (! (m_str = (char*)malloc (strlen (str ? str : "") + 1)))
            throw MemoryException (strerror (errno));
        ... }
};
```

  - try { String str (...); ... }

```
try { String str (...); ... }
catch (MemoryException& ex) { ... }
```

知识讲解

不完整对象的回滚

Tarena Technology 达内科技

知识讲解

- 构造函数抛出异常，对象将被不完整构造，而一个被不完整构造的对象，其析构函数永远不会被执行
  - 构造函数的回滚机制，可以保证所有成员子对象和基类子对象，在抛出异常的过程中，都能得到正确地析构
  - 所有动态分配的资源，必须在抛出异常之前，手动释放，否则将形成内存泄漏，除非使用了智能指针
  - if (抛出异常的条件满足) {

```
    释放此前动态分配的资源;
    throw 异常对象;
}
```

构造过程中的异常

【参见：TTS COOKBOOK 】

- 构造过程中的异常

Technology Tarena 达内科技

析构函数抛出异常

Tarena Technology 达内科技

知识讲解

- 不要从析构函数中主动抛出异常
  - 在两种情况下，析构函数会被调用
    1. 正常销毁对象，离开作用域或显式delete
    2. 在异常传递的堆栈辗转开解(stack-unwinding)过程中，由异常处理系统销毁对象
  - 对于第二种情况，异常正处于激活状态，而析构函数又抛出了异常，并试图将流程移至析构函数之外，这时C++将通过std::terminate()函数，令进程中止

尽量内部消化异常

Tarena Technology 达内科技

知识讲解

- 在析构函数中，执行任何可能引发异常的操作，都尽量把异常在内部消化掉，防止其从析构函数中被继续抛出
  - Dummy::~Dummy (void) {

```
Dummy::~Dummy (void) {
    try {
        ...
    }
    catch (FileException& ex) { ... }
    catch (MemoryException& ex) { ... }
    catch (GeneralException& ex) { ... }
    catch (runtime_error& ex) { ... }
    catch (exception& ex) { ... }
    catch (...) {}
}
```

析构过程中的异常

【参见：TTS COOKBOOK 】

- 析构过程中的异常

标准异常

Technology Tarena 达内科技

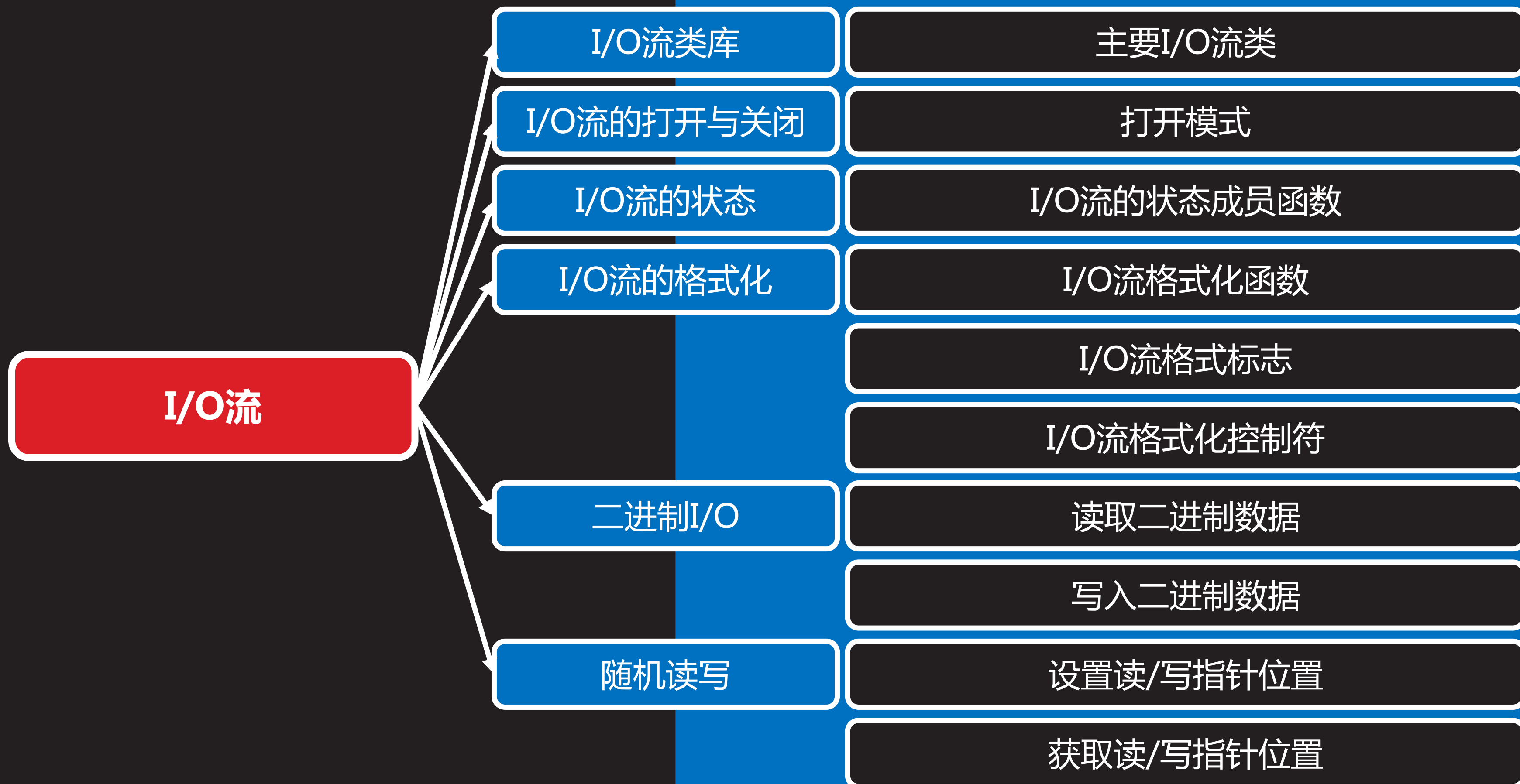
标准异常

- 标准C++库中预定义的异常类型

```
#include <stdexcept>
```

- exception
  - runtime_error
    - overflow_error
    - underflow_error
  - logic_error
    - invalid_argument
    - length_error
    - out_of_range
  - bad_alloc
  - bad_cast
  - bad_type_id
  - bad_exception

I/O流



Tarena Technology 达内科技

Technology
Tarena
达内科技

主要I/O流类

- I/O流类系

ios_base
ios
streambuf
conbuf
filebuf
strstreambuf
fstreambase
istream
ostream
strstreambase
ifstream
ofstream
iostream
istrstream
ostrstream
fstream
istream_withassign
ostream_withassign
strstream
iostream_withassign

Technology Tarena 达内科技

打开模式

Tarena Technology 达内科技

知识讲解

- ios::in
  - 打开文件用于读取，不存在则失败，存在不清空
  - 适用于ifstream(缺省)/fstream
- ios::out
  - 打开文件用于写入，不存在则创建，存在则清空
  - 适用于ofstream(缺省)/fstream
- ios::app
  - 打开文件用于追加，不存在则创建，存在不清空
  - 适用于ofstream/fstream

打开模式（续1）

Tarena Technology 达内科技

知识讲解

- ios::trunc
  - 打开时清空原内容
  - 适用于ofstream/fstream
- ios::ate
  - 打开时定位文件尾
  - 适用于ifstream/ofstream/fstream
- ios::binary
  - 以二进制模式读写
  - 适用于ifstream/ofstream/fstream

打开模式（续2）

Tarena Technology 达内科技

知识讲解

- 打开模式可以组合使用
  - ios::in | ios::out
    表示既读取又写入
- 打开模式不能随意组合
  - ios::in | ios::trunc
    清空同时读取没有意义
  - ios::in | ios::out | ios::trunc
    合理，清空原内容，写入新内容，同时读取

I/O流的状态

I/O流的状态成员函数

Tarena Technology 达内科技

知识讲解

状态成员函数	说明
bool ios::good (void);	流可用，状态位全零，返回true
bool ios::bad (void);	badbit位被设置否
bool ios::eof (void);	eofbit位被设置否
bool ios::fail (void);	badbit或failbit位被设置否
iostate ios::rdstate (void);	获取当前状态
void ios::clear (iostate s = ios::goodbit);	设置(复位)流状态
void ios::setstate (iostate s);	添加流状态

I/O流的格式化

I/O流格式化函数

- I/O流类(ios)定义了一组用于控制输入输出格式的公有成员函数，调用这些函数可以改变I/O流对象内部的格式状态，进而影响后续输入输出的格式化方式

知识讲解

格式化函数	说明
int ios::precision (int);	设置浮点精度，返回原精度
int ios::precision (void) const;	获取浮点精度
int ios::width (int);	设置显示域宽，返回原域宽
int ios::width (void) const;	获取显示域宽
char ios::fill (char);	设置填充字符，返回原字符
char ios::fill (void) const;	获取填充字符
long ios::flags (long);	设置格式标志，返回原标志
long ios::flags (void) const;	获取格式标志

I/O流格式化函数（续1）

Tarena Technology 达内科技

知识讲解

格式化函数	说明
long ios::setf (long);	添加格式标志位，返回原标志
long ios::setf (long, long);	添加格式标志位，返回原标志 先用第二个参数将互斥域清零
long ios::unsetf (long);	清除格式标志位，返回原标志

- 一般而言，对I/O流格式的改变都是持久的，即只要不再设置新格式，当前格式将始终保持下去
- 显示域宽是个例外，通过ios::width(int)所设置的显示域宽，只影响紧随其后的第一次输出，再往后的输出又恢复到默认状态

I/O流格式标志

Technology Tarena 达内科技

知识讲解

格式标志位	互斥域	说明
ios::left	ios::adjustfield	左对齐
ios::right		右对齐
ios::internal		数值右对齐，符号左对齐
ios::dec	ios::basefield	十进制
ios::oct		八进制
ios::hex		十六进制
ios::fixed	ios::floatfield	用定点小数表示浮点数
ios::scientific		用科学计数法表示浮点数

I/O流格式标志（续1）

Tarena Technology 达内科技

知识讲解

格式标志位	说明
ios::showpos	正整数前面显示+号
ios::showbase	显示进制前缀0或0x
ios::showpoint	显示小数点和尾数0
ios::uppercase	数中字母显示为大写
ios::boolalpha	用字符串表示布尔值
ios::unitbuf	每次插入都刷流缓冲
ios::skipws	以空白字符作分隔符

I/O流格式化控制符

Tarena Technology 达内科技

知识讲解

格式化控制符	说明	输入	输出
left	左对齐	☐	☑
right	右对齐	☐	☑
internal	数值右对齐，符号左对齐	☐	☑
dec	十进制	☑	☑
oct	八进制	☑	☑
hex	十六进制	☑	☑
fixed	用定点小数表示浮点数	☐	☑
scientific	用科学计数法表示浮点数	☐	☑

I/O流格式化控制符（续1）

Tarena Technology 达内科技

知识讲解

格式化控制符	说明	输入	输出
(no)showpos	正整数前面(不)显示+号	☐	☑
(no)showbase	(不)显示进制前缀0或0x	☐	☑
(no)showpoint	(不)显示小数点和尾数0	☐	☑
(no)uppercase	数中字母(不)显示为大写	☐	☑
(no)boolalpha	(不)用字符串表示布尔值	☑	☑
(no)unitbuf	(不)每次插入都刷流缓冲	☐	☑
(no)skipws	(不)以空白字符作分隔符	☑	☐
ws	跳过前导空白字符	☑	☐

I/O流格式化控制符（续2）

Tarena Technology 达内科技

知识讲解

格式化控制符	说明	输入	输出
ends	空字符	☐	☑
endl	换行符，刷流缓冲	☐	☑
flush	刷流缓冲	☐	☑
setprecision (int)	设置浮点精度	☐	☑
setw (int)	设置显示域宽	☐	☑
setfill (int)	设置填充字符	☐	☑
setiosflags (long)	设置格式标志	☑	☑
resetiosflags (long)	清除格式标志	☑	☑

I/O流的格式化

【参见：TTS COOKBOOK 】

- I/O流的格式化

二进制I/O

读取二进制数据

Tarena Technology 达内科技

知识讲解

- istream& istream::read (char* buffer,
  streamsize num);
  - 从输入流中读取num个字节到缓冲区buffer中
  - 返回流本身，其在布尔上下文中的值，成功(读满)为true，失败(没读满)为false
  - 如果没读满num个字节，函数就返回了，比如遇到文件尾，最后一次读到缓冲区buffer中的字节数，可以通过istream::gcount()函数获得

写入二进制数据

- ostream& ostream::write (const char* buffer, streamsize num);
  - 将缓冲区buffer中的num个字节写入到输出流中
  - 返回流本身，其在布尔上下文中的值，成功(写满)为true，失败(没写满)为false

知识讲解

文件复制

【参见：TTS COOKBOOK 】

- 文件复制

Technology
Tarena
达内科技

设置读/写指针位置

Tarena Technology 达内科技

知识讲解

- istream& istream::seekg (off_type offset,
    ios::seekdir origin);
  ostream& ostream::seekp (off_type offset,
    ios::seekdir origin);
  - origin表示偏移量offset的起点
    ios::beg：从文件的第一个字节
    ios::cur：从文件的当前位置
    ios::end：从文件最后一个字节的下一个位置
  - offset为负/正表示向文件头/尾的方向偏移
  - 读/写指针被移到文件头之前或文件尾之后，则失败

获取读/写指针位置

Tarena Technology 达内科技

知识讲解

- pos_type istream::tellg (void);
  pos_type ostream::tellp (void);
  – 返回读/写指针当前位置相对于文件头的字节偏移量
- iostream的子类，如fstream
  – 同时拥有针对读/写指针位置的两套设置/获取函数
  – 理论上应该拥有两个相互独立的读/写指针
  – 多数编译器仍然使用一个指针记录文件当前位置
  – 建议读取时用seekg/tellg，写入时用seekp/tellp

随机读写

【参见：TTS COOKBOOK 】

课堂练习

- 随机读写

总结和答疑