choose函数

1. **概述**

头文件 choose.hpp

源文件 choose.cpp

1. **可变参数表风格的 choose 函数**

**函数原型：**

**template** **<class** **T>**

**T** **choose(int** index**,** **T** value0**,** **...)** **throw** **(invalid\_argument);**

**用法示例：**

**/\*在 C++11 下调试, typeid(result1).name() 方法用来输出返回值的类型\*/**

**auto** result1 **=** choose**(***2***,** **'a',** **'b',** **'c',** **'d');** **//返回 char 型 'c'**

**auto** result2 **=** choose**(***3***,** *1***,** *2***,** *3***,** *4***);** **//返回 int 型 4**

**auto** result3 **=** choose**(***3***,** *1.1***,** *2.2***,** *3.3***,** *4.4***);** **//返回 double 型 4.4**

**auto** result4 **=** choose**(***2***,** **"abc",** **"def",** **"ghi");** **//返回 char\* 型 "ghi"**

**cout** **<<** **typeid(**result1**).**name**()** **<<** **" "** **<<** result1 **<<** **endl;//输出 c c**

**cout** **<<** **typeid(**result2**).**name**()** **<<** **" "** **<<** result2 **<<** **endl;//输出 i 4**

**cout** **<<** **typeid(**result3**).**name**()** **<<** **" "** **<<** result3 **<<** **endl;//输出 d 4.4**

**cout** **<<** **typeid(**result4**).**name**()** **<<** **" "** **<<** result4 **<<** **endl;//输出 PKc ghi**

**/\*在 C++0x 下调试, 与上一种写法逻辑上等价\*/**

**cout** **<<** **typeid(**choose**(***2***,** **'a',** **'b',** **'c',** **'d')).**name**()** **<<** **" "** **<<** choose**(***2***,** **'a',** **'b',** **'c',** **'d')** **<<** **endl;** **//输出 c c**

**cout** **<<** **typeid(**choose**(***3***,** *1***,** *2***,** *3***,** *4***)).**name**()** **<<** **" "** **<<**

choose**(***3***,** *1***,** *2***,** *3***,** *4***)** **<<** **endl;** **//输出 i 4**

**cout** **<<** **typeid(**choose**(***3***,** *1.1***,** *2.2***,** *3.3***,** *4.4***)).**name**()** **<<** **" "** **<<** choose**(***3***,** *1.1***,** *2.2***,** *3.3***,** *4.4***)** **<<** **endl;** **//输出 d 4.4**

**cout** **<<** **typeid(**choose**(***2***,** **"abc",** **"def",** **"ghi")).**name**()** **<<** **" "** **<<** choose**(***2***,** **"abc",** **"def",** **"ghi")** **<<** **endl;** **//输出 PKc ghi**

1. **初始化列表风格的 choose 函数**

**函数原型：**

**template** **<class** **T>**

**T** **choose(int** index**,** **initializer\_list<T>** src**)** **throw** **(invalid\_argument,** **out\_of\_range);**

**用法示例：**

**/\*在 C++11 下调试, typeid(result1).name() 方法用来输出返回值的类型\*/**

**auto** result1 **=** choose**(***2***,** **{** **'a',** **'b',** **'c',** **'d'** **});** **//返回 char 型 'c'**

**auto** result2 **=** choose**(***3***,** **{** *1***,** *2***,** *3***,** *4* **});** **//返回 int 型 4**

**auto** result3 **=** choose**(***3***,** **{** *1.1***,** *2.2***,** *3.3***,** *4.4* **});** **//返回 double 型 4.4**

**auto** result4 **=** choose**(***2***,** **{** **"abc",** **"def",** **"ghi"** **});** **//返回 char\* 型 "ghi"**

**cout** **<<** **typeid(**result1**).**name**()** **<<** **" "** **<<** result1 **<<** **endl;** **//输出 c c**

**cout** **<<** **typeid(**result2**).**name**()** **<<** **" "** **<<** result2 **<<** **endl;** **//输出 i 4**

**cout** **<<** **typeid(**result3**).**name**()** **<<** **" "** **<<** result3 **<<** **endl;** **//输出 d 4.4**

**cout** **<<** **typeid(**result4**).**name**()** **<<** **" "** **<<** result4 **<<** **endl;** **//输出 PKc ghi**

**/\*在 C++0x 下调试, 与上一种写法逻辑上等价\*/**

**cout** **<<** **typeid(**choose**(***2***,** **{** **'a',** **'b',** **'c',** **'d'** **})).**name**()** **<<** **" "** **<<** choose**(***2***,** **{** **'a',** **'b',** **'c',** **'d'** **})** **<<** **endl;** **//输出 c c**

**cout** **<<** **typeid(**choose**(***3***,** **{** *1***,** *2***,** *3***,** *4* **})).**name**()** **<<** **" "** **<<**

choose**(***3***,** **{** *1***,** *2***,** *3***,** *4* **})** **<<** **endl;** **//输出 i 4**

**cout** **<<** **typeid(**choose**(***3***,** **{** *1.1***,** *2.2***,** *3.3***,** *4.4* **})).**name**()** **<<** **" "** **<<** choose**(***3***,** **{** *1.1***,** *2.2***,** *3.3***,** *4.4* **})** **<<** **endl;** **//输出 d 4.4**

**cout** **<<** **typeid(**choose**(***2***,** **{** **"abc",** **"def",** **"ghi"** **})).**name**()** **<<** **" "** **<<** choose**(***2***,** **{** **"abc",** **"def",** **"ghi"** **})** **<<** **endl;** **//输出 PKc ghi**

1. **可变参数模板风格的 choose 函数**

**函数原型：**

**template** **<class** **T,** **class** **...Args>**

**T** **choose\_c11(int** index**,** **T** head**,** **Args** **...** args**)** **throw** **(invalid\_argument,** **out\_of\_range);**

**解包函数与递归终止函数原型：**

**namespace** **//匿名命名空间内的内容对文件外不可见**

**{**

**//递归终止函数**

**int** **choose\_unpack(int** total**);**

**//展开函数**

**template** **<class** **T,** **class** **...Args>**

**T** **choose\_unpack(int** total**,** **T** head**,** **Args** **...** rest**);**

**}**

**三种风格choose函数的比较**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 可变参数表风格 | 初始化列表风格 | 可变参数模板风格 |
| 正确用法示例 | choose(2, 'a', 'b', 'c', 'd')  choose(3, 1, 2, 3, 4)  choose(3, 1.1, 2.2, 3.3, 4.4)  choose(2, "abc", "def", "ghi") | choose(2, { 'a', 'b', 'c', 'd' })  choose(3, { 1, 2, 3, 4 })  choose(3, { 1.1, 2.2, 3.3, 4.4 })  choose(2, {"abc","def","ghi" }) | choose\_c11(2, 'a', 'b', 'c', 'd')  choose\_c11(3, 1, 2, 3, 4)  choose\_c11(3, 1.1, 2.2, 3.3, 4.4)  choose\_c11(2, "abc", "def", "ghi") |
| C++0x支持 | 是 | 否 | 否 |
| C++11支持 | 是 | 是 | 是 |
| 索引值为负 | choose(-1, 'a', 'b', 'c', 'd')  抛出**invalid\_argument**异常 | choose(-1, { 'a', 'b', 'c', 'd' })  抛出**invalid\_argument**异常 | choose\_c11(-1, 'a', 'b', 'c', 'd')  抛出**invalid\_argument**异常 |
| 越界访问 | choose(4, 'a', 'b', 'c', 'd')  返回不确定值 | choose(4, { 'a', 'b', 'c', 'd' })  抛出**out\_of\_range**异常 | choose\_c11(4, 'a', 'b', 'c', 'd')  抛出**out\_of\_range**异常 |
| 项目类型不统一 | choose(2, 'a', 'b', 65, 'd')  有返回无意义值的风险 | choose(2, { 'a', 'b', 65, 'd' })  编译错误 | choose\_c11(2, 'a', 'b', 65, 'd')  按第一个项目的类型返回, 即‘A‘ |
| 隐式类型转换 | choose(3, 1, 65.2, 3, 4)  有返回无意义值的风险 | choose(3, { 1, 65.2, 3, 4 })  编译错误 | choose\_c11(3, 0.0, 1.1, 2, 3.3)  按所有排在前面的项目中精度最低的类型转换后，再按第一个项目的类型返回, 即3.0(double) |
| 优点 | C++11之前的版本支持 | 越界访问检查  项目类型不统一时会检查出来  选择速度较快 | 越界访问检查  允许项目的类型不统一 |
| 缺点 | 无越界访问检查  项目类型不统一的风险  隐式类型转化的风险 | C++11之前的版本不支持  不允许项目的类型不统一 | C++11之前的版本不支持  会触发类型转换, 发生类型间无法转换的错误或者损失精度  需要解包函数递归调用找到待选项目, 速度较慢 |