|  |  |
| --- | --- |
| 1. 重载与模板 2. 编写重载模板 3. 多个可行模板 4. 非模板和模板重载 5. 重载模板和类型转换 6. 缺少声明可能导致程序行为异常 | Template <typename T> string debug\_rep(const T &t)  Template <typename T> string debug\_rep(T \*p)  String s(“hi”)  (1)Debug\_rep(s)  (2)Debug\_rep(&s)  Const string \*sp = &s;  (3)Debug\_rep(sp);  String debug\_rep(const string &s);  Debug\_rep(“hi wolrd”)  Char \*p  (4)Debug\_rep(string(p)) |
| 1. 重载与模板      1. 函数模板可以被另一个模板或一个普通非模板函数重载。名字相同的函数必须具有不同数量或类型的参数 2. 编写重载模板 3. String s(“hi”)   Debug\_rep(s);  只有第一个版本的debug\_rep可行。第二个版本不行，要求指针实参。   1. Debug\_rep(&s)   生成：Debug\_rep(const string\*&),T被绑定到string\*  Debug\_rep(string\*),T被绑定到string  第二个版本精确匹配，第一个版本需要进行普通指针到const指针的转换   1. 多个可行模板 2. Const string \*sp = &s;   Debug\_rep(sp);  Debug\_rep(const string\*&),T绑定到string\*  Debug\_rep(const string\*),T绑定到cosnt string  还是第二个模板更精确。   1. 规则：无法对一个const的指针调用指针版本的debug\_rep 2. note:匹配规则4.2 3. 非模板和模板重载   String debug\_rep(const string &)   1. String s(“hi”); debug\_rep(s)   第一个模板可以T被绑定string  普通非模板参数也可以   1. 提供同样好的匹配，编译器会选择非模板版本。编译器选择最特例化的版本 2. Note:匹配规则4.1 3. 重载模板和类型转换 4. C风格字符串指针和字符串字面常量.debug\_rep(“hasdf”); 5. 前两个模板提供精确匹配   Debug\_rep(const T&) T被绑定到char[10]  Debug\_rep(T\*),T被绑定到const char  编译器选择第二个模板T\*更加特例化。   1. Debug\_rep(string(p))，   需要一次用户定义的类型转换才能精确匹配这个  Debug\_rep(const string&)   1. 缺少声明可能导致程序行为异常 2. 若没有声明函数，重载函数模板会从模板实例化出与调用匹配的版本 3. 若不声明所有函数，编译器调用的函数可能与所需的版本不一样 | |