بازتاب و متاکلاس‌ها در C++23

Table of Contents

بازتاب و متاکلاس‌ها در ++C23: طراحی و پیاده‌سازی

مطالعه جامع بر روی سیستم بازتاب و الگوی متاکلاس در استاندارد جدید ++C

نویسنده اول، نویسنده دوم

دانشگاه علوم کامپیوتر، انستیتوی تحقیقات فناوری

چکیده

این مقاله به بررسی جامع سیستم بازتاب (Reflection) و الگوی متاکلاس (Metaclass) در استاندارد جدید ++C23 می‌پردازد. با معرفی قابلیت‌های جدید زبان، این تحقیق راه‌حل‌های نوآورانه‌ای برای مشکلات برنامه‌نویسی متاپروگرمینگ ارائه می‌دهد.

کلمات کلیدی: بازتاب، متاکلاس، ++C23، متاپروگرمینگ، کامپایل تایم

ΓÇö title: ΓÇ£╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪»╪▒ C++23: ╪ó█î┘å╪»┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█îΓÇ¥ subtitle: ΓÇ£╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ΓÇ¥ author: ΓÇ£┘à╪¡┘à╪»╪▒╪╢╪º ╪╣┘ä█î┘╛┘ê╪▒ΓÇ¥ email: ΓÇ£mamarezaalipour@gmail.comΓÇ¥ date: ΓÇ£╪ó┌»┘ê╪│╪¬ 2025ΓÇ¥ journal: ΓÇ£┘à╪¼┘ä┘ç ╪¬╪¡┘é█î┘é╪º╪¬ ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ΓÇ¥ documentclass: article geometry: margin=2.5cm fontsize: 12pt linestretch: 1.5 header-includes: -

* - -
  + - - - -
    - * ΓÇö # ┌å┌⌐█î╪»┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» C++23 ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘╛█î╪┤┌»╪º┘à╪º┘å┘ç ┘ê ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪▒╪º ┘à╪╣╪▒┘ü█î ┘à█î┌⌐┘å╪» ┌⌐┘ç ┌å╪┤┘à╪º┘å╪»╪º╪▓ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ╪¿┘å█î╪º╪»█î ╪¬╪║█î█î╪▒ ┘à█î╪»┘ç╪». ╪º█î┘å ┘à┘é╪º┘ä┘ç ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣█î ╪º╪▓ ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¼╪»█î╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» ┘ê ┘à╪¿╪º┘å█î ┘å╪╕╪▒█î╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪╣┘à┘ä█î ┘ê ╪¬╪ú╪½█î╪▒╪º╪¬ ╪╣┘à┘ä┌⌐╪▒╪»█î ╪ó┘å┘ç╪º ╪▒╪º ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ╪▒┘ê╪┤┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ╪º╪▓ ╪╖╪▒█î┘é ╪º╪▒╪▓█î╪º╪¿█î ┌»╪│╪¬╪▒╪»┘ç ╪╣┘à┘ä┌⌐╪▒╪» ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î╪î ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç█î┘à ┌⌐┘ç ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪»╪▒ ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä (┌⌐╪º┘ç╪┤ ╪¬╪º 40% ╪»╪▒ ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç)╪î ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ┌⌐╪» (┌⌐╪º┘ç╪┤ 60-80% ┌⌐╪» ╪║█î╪▒╪╢╪▒┘ê╪▒█î) ┘ê ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪»╪▒ ╪¡█î┘å ╪¡┘ü╪╕ ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿┘ç ╪»╪│╪¬ ┘à█î╪ó┘ê╪▒┘å╪». ╪¬╪¡┘é█î┘é ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ╪▒╪º█î╪¼╪î ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ê ╪▓╪¿╪º┘å┘ç╪º█î ╪«╪º╪╡ ╪»╪º┘à┘å┘ç ╪▒╪º ╪¿╪º ╪│┘ç┘ê┘ä╪¬ ┘ê ┌⌐╪º╪▒╪º█î█î ╪¿█î╪│╪º╪¿┘é┘ç ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å┘å╪». ╪º█î┘å ┘à╪╖╪º┘ä╪╣┘ç ╪┤╪º┘à┘ä ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ╪»╪▒ ╪│┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º╪╡┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ (GCC 13╪î Clang 16╪î MSVC 2023) ╪º╪│╪¬ ┘ê ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪¼╪»█î╪» ╪»╪▒ ╪¬┘ê┘ä█î╪» ORM ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç╪î ╪¬┘ê╪│╪╣┘ç ┌å╪º╪▒┌å┘ê╪¿ GUI ┘ê ╪▓█î╪▒╪│╪º╪«╪¬ ╪¬╪│╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». **┌⌐┘ä┘à╪º╪¬ ┌⌐┘ä█î╪»█î:** ╪¿╪º╪▓╪¬╪º╪¿ C++23╪î ┘à╪¬╪º┌⌐┘ä╪º╪│╪î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î╪î ╪¬┘ê┘ä█î╪» ┌⌐╪» ΓÇö # ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪»╪▒ C++23: ╪ó█î┘å╪»┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ## ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ **┘å┘ê█î╪│┘å╪»┌»╪º┘å:** ┘à╪¡┘à╪»╪▒╪╢╪º ╪╣┘ä█î┘╛┘ê╪▒ (mamarezaalipour@gmail.com) **┌⌐┘ä█î╪»┘ê╪º┌ÿ┘ç┘ç╪º:** C++23╪î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐╪î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º╪î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î╪î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿╪î ╪¬┘ê┘ä█î╪» ┌⌐╪» ΓÇö ## ┌å┌⌐█î╪»┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» C++23 ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘╛█î╪┤┌»╪º┘à╪º┘å┘ç ┘ê ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪▒╪º ┘à╪╣╪▒┘ü█î ┘à█î┌⌐┘å╪» ┌⌐┘ç ┌å╪┤┘à╪º┘å╪»╪º╪▓ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ╪¿┘å█î╪º╪»█î ╪¬╪║█î█î╪▒ ┘à█î╪»┘ç╪». ╪º█î┘å ┘à┘é╪º┘ä┘ç ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣█î ╪º╪▓ ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¼╪»█î╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» ┘ê ┘à╪¿╪º┘å█î ┘å╪╕╪▒█î╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪╣┘à┘ä█î ┘ê ╪¬╪ú╪½█î╪▒╪º╪¬ ╪╣┘à┘ä┌⌐╪▒╪»█î ╪ó┘å┘ç╪º ╪▒╪º ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ╪▒┘ê╪┤┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ╪º╪▓ ╪╖╪▒█î┘é ╪º╪▒╪▓█î╪º╪¿█î ┌»╪│╪¬╪▒╪»┘ç ╪╣┘à┘ä┌⌐╪▒╪» ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î╪î ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç█î┘à ┌⌐┘ç ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪»╪▒ ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä (┌⌐╪º┘ç╪┤ ╪¬╪º 40% ╪»╪▒ ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç)╪î ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ┌⌐╪» (┌⌐╪º┘ç╪┤ 60-80% ┌⌐╪» ╪║█î╪▒╪╢╪▒┘ê╪▒█î) ┘ê ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪»╪▒ ╪¡█î┘å ╪¡┘ü╪╕ ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿┘ç ╪»╪│╪¬ ┘à█î╪ó┘ê╪▒┘å╪». ╪¬╪¡┘é█î┘é ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ╪▒╪º█î╪¼╪î ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ê ╪▓╪¿╪º┘å┘ç╪º█î ╪«╪º╪╡ ╪»╪º┘à┘å┘ç ╪▒╪º ╪¿╪º ╪│┘ç┘ê┘ä╪¬ ┘ê ┌⌐╪º╪▒╪º█î█î ╪¿█î╪│╪º╪¿┘é┘ç ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å┘å╪». ╪º█î┘å ┘à╪╖╪º┘ä╪╣┘ç ╪┤╪º┘à┘ä ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ╪»╪▒ ╪│┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º╪╡┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ (GCC 13╪î Clang 16╪î MSVC 2023) ╪º╪│╪¬ ┘ê ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪¼╪»█î╪» ╪»╪▒ ╪¬┘ê┘ä█î╪» ORM ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç╪î ╪¬┘ê╪│╪╣┘ç ┌å╪º╪▒┌å┘ê╪¿ GUI ┘ê ╪▓█î╪▒╪│╪º╪«╪¬ ╪¬╪│╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». ╪º█î┘å █î╪º┘ü╪¬┘ç┘ç╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç ╪¬╪║█î█î╪▒ ┘╛╪º╪▒╪º╪»╪º█î┘à ╪¿┘ç ╪│┘à╪¬ ┌⌐╪» C++ ╪º╪╣┘ä╪º┘å█î╪¬╪▒ ┘ê ┘é╪º╪¿┘ä ┘å┌»┘ç╪»╪º╪▒█î╪¬╪▒ ╪º╪│╪¬ ┘ê ╪▓╪¿╪º┘å ╪▒╪º ╪»╪▒ ┘à┘é╪º╪¿┘ä ╪¼╪º█î┌»╪▓█î┘å┘ç╪º█î ┘à╪»╪▒┘å ┘à┘ê┘é╪╣█î╪¬ ╪▒┘é╪º╪¿╪¬█î ┘à█î╪»┘ç╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪»█î ┘ê ╪│╪º╪▓┌»╪º╪▒█î ┘╛╪│╪▒┘ê ╪ó┘å ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å╪». ΓÇö ## 1. ┘à┘é╪»┘à┘ç ### 1.1 ╪¬┌⌐╪º┘à┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ C++ ┘à╪»╪¬┘ç╪º╪│╪¬ ┌⌐┘ç ╪¿┘ç ╪«╪º╪╖╪▒ ╪│█î╪│╪¬┘à ┘é╪º┘ä╪¿ ┘é╪»╪▒╪¬┘à┘å╪»╪┤ ╪┤┘å╪º╪«╪¬┘ç ┘à█î╪┤┘ê╪» ┌⌐┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘╛█î┌å█î╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪º╪▓ ╪▓┘à╪º┘å ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î C++98 ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪»┘ç ╪º╪│╪¬. ╪¿╪º ╪º█î┘å ╪¡╪º┘ä╪î ╪│┘ü╪▒ ╪º╪▓ ╪¬╪«╪╡╪╡█î╪º╪¿█î ╪º┘ê┘ä█î┘ç ┘é╪º┘ä╪¿ ╪¿┘ç ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç╪º█î ┌⌐┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î ┘à╪»╪▒┘å C++ ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î╪┤┘ê╪»╪î ┘ç┘à ┘å╪¿┘ê╪║ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç ╪¼╪º┘à╪╣┘ç C++ ┘ê ┘ç┘à ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪¿┘å█î╪º╪»█î ╪▒┘ê█î┌⌐╪▒╪» ┘ü╪╣┘ä█î ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [1╪î 2]. ╪│█î╪│╪¬┘à ┘é╪º┘ä╪¿ ┌⌐┘ç ╪»╪▒ ╪º╪¿╪¬╪»╪º ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪╖╪▒╪º╪¡█î ╪┤╪»┘ç ╪¿┘ê╪»╪î ╪¿╪│█î╪º╪▒ ┘ü╪▒╪º╪¬╪▒ ╪º╪▓ ┘à╪¡╪»┘ê╪»┘ç ╪º┘ê┘ä█î┘ç ╪«┘ê╪» ┌⌐╪┤█î╪»┘ç ╪┤╪»┘ç ╪¬╪º ╪¿┘ç ╪╣┘å┘ê╪º┘å █î┌⌐ ╪│█î╪│╪¬┘à ┘à╪¡╪º╪│╪¿┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¬┘ê╪▒█î┘å┌»-┌⌐╪º┘à┘ä ╪╣┘à┘ä ┌⌐┘å╪» [3]. ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î█î ┘à╪º┘å┘å╪» Boost.MPL [4]╪î Boost.Hana [5] ┘ê <type\_traits> ╪«┘ê╪» ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ┘é╪»╪▒╪¬ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪»╪î ╪º┘à╪º ┘ç┘à┌å┘å█î┘å ┘╛█î┌å█î╪»┌»█î ╪░╪º╪¬█î ┘ê ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î ╪┤█î╪¿╪»╪º╪▒ ╪ó┘å ╪▒╪º ╪¿╪▒╪¼╪│╪¬┘ç ┘à█î┌⌐┘å┘å╪» [6]. ╪¬┌⌐╪º┘à┘ä █î┌⌐ ┘à┘ü┘ç┘ê┘à ╪│╪º╪»┘ç ╪▒╪º ╪»╪▒ ┘å╪╕╪▒ ╪¿┌»█î╪▒█î╪»: ╪¬┌⌐╪▒╪º╪▒ ╪▒┘ê█î ╪º╪╣╪╢╪º█î █î┌⌐ struct. ╪»╪▒ C++ ╪│┘å╪¬█î╪î ╪º█î┘å ╪º┘à╪▒ ┘å█î╪º╪▓ ╪¿┘ç ┘à╪º╪┤█î┘å╪ó┘ä╪º╪¬ ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç╪î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î SFINAE ┘ê ╪º╪║┘ä╪¿ ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪«╪º╪▒╪¼█î ╪»╪º╪▒╪» [7]. ┘╛█î╪┤╪▒┘ü╪¬ ╪▓█î╪▒ ╪º█î┘å ╪¬┌⌐╪º┘à┘ä ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪»: cpp // C++98: ╪¬╪«╪╡╪╡█î╪º╪¿█î ╪»╪│╪¬█î ╪¿╪▒╪º█î ┘ç╪▒ ┘å┘ê╪╣ template<> struct serializer<Person> { /\* ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪»╪│╪¬█î \*/ }; // C++11: ╪┤┘å╪º╪│╪º█î█î ┘à╪¿╪¬┘å█î ╪¿╪▒ SFINAE template<typename T, typename = void> struct has\_serialize : std::false\_type {}; template<typename T> struct has\_serialize<T, std::void\_t<decltype(std::declval<T>().serialize())>> : std::true\_type {}; // C++20: ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ╪¿╪▒╪º█î ┘å╪¡┘ê ┘╛╪º┌⌐╪¬╪▒ template<typename T> concept Serializable = requires(T t) { t.serialize(); }; // C++23: ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪│╪¬┘é█î┘à template<typename T> void serialize(const T& obj) { constexpr auto members = std::meta::data\_members\_of(std::meta::reflexpr(T)); // ╪¬┌⌐╪▒╪º╪▒ ┘à╪│╪¬┘é█î┘à ╪▒┘ê█î ╪º╪╣╪╢╪º█î ┘ê╪º┘é╪╣█î } ╪º█î┘å ┘╛█î╪┤╪▒┘ü╪¬ ┘å┘ç ╪¬┘å┘ç╪º ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘å╪¡┘ê█î ╪¿┘ä┌⌐┘ç ╪¬╪║█î█î╪▒╪º╪¬ ╪¿┘å█î╪º╪»█î ╪»╪▒ ┘é╪º╪¿┘ä█î╪¬ ╪¿█î╪º┘å ┘ê ┘å┌»┘ç╪»╪º╪▒█î ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [8]. ### 1.2 ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪▒┘ê╪┤┘ç╪º█î ┘ü╪╣┘ä█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ╪╣┘ä█î╪▒╪║┘à ┘é╪»╪▒╪¬ ╪ó┘å┘ç╪º╪î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘ü╪╣┘ä█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪º╪▓ ┌å┘å╪»█î┘å ┘à╪¡╪»┘ê╪»█î╪¬ ╪¡█î╪º╪¬█î ╪▒┘å╪¼ ┘à█î╪¿╪▒┘å╪» ┌⌐┘ç ╪º╪▓ ┘╛╪░█î╪▒╪┤ ┘ê ╪º╪½╪▒╪¿╪«╪┤█î ╪ó┘å┘ç╪º ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ╪¿╪▓╪▒┌» ┘à┘é█î╪º╪│ ╪¼┘ä┘ê┌»█î╪▒█î ┘à█î┌⌐┘å╪» [9╪î 10]: **┘╛█î┌å█î╪»┌»█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä:** ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ┘é╪º┘ä╪¿ ╪º╪▓ ╪º┘ä┌»┘ê█î ╪▒╪┤╪» ┘å┘à╪º█î█î ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┘╛█î┌å█î╪»┘ç ┘╛█î╪▒┘ê█î ┘à█î┌⌐┘å╪». ┘à╪╖╪º┘ä╪╣╪º╪¬ ╪º┘ê┘ä█î┘ç ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ┘╛╪º█î┌»╪º┘ç┘ç╪º█î ┌⌐╪» ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ ┘à█î╪¬┘ê╪º┘å┘å╪» ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¬╪¼╪▒╪¿┘ç ┌⌐┘å┘å╪» ┌⌐┘ç ╪¿┘ç ╪╣┘å┘ê╪º┘å O(n┬▓) █î╪º ╪¿╪»╪¬╪▒ ╪¿╪º ╪¬╪╣╪»╪º╪» ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î ┘é╪º┘ä╪¿ ┘ê ╪¬╪«╪╡╪╡█î╪º╪¿█î┘ç╪º ┘à┘é█î╪º╪│ ┘à█î╪┤┘ê╪» [11]. ╪│█î╪│╪¬┘à┘ç╪º█î ╪│╪º╪«╪¬ ┘à╪»╪▒┘å ╪¿╪º ╪º█î┘å ┘╛█î┌å█î╪»┌»█î ┘à╪¿╪º╪▒╪▓┘ç ┘à█î┌⌐┘å┘å╪»╪î ╪¿┘ç ┘ê█î┌ÿ┘ç ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪¬╪»╪▒█î╪¼█î. **┌⌐█î┘ü█î╪¬ ┘╛█î╪º┘à ╪«╪╖╪º:** ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º█î ┘é╪º┘ä╪¿ ╪¿╪»┘å╪º┘à ┘ç╪│╪¬┘å╪» ┌⌐┘ç ╪¬┘ü╪│█î╪▒ ╪ó┘å┘ç╪º ╪»╪┤┘ê╪º╪▒ ╪º╪│╪¬ ┘ê ╪º╪║┘ä╪¿ ╪╡╪»┘ç╪º ╪«╪╖ ╪«╪▒┘ê╪¼█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¿╪º ╪º╪▒╪¼╪º╪╣╪º╪¬ ╪▒┘à╪▓╪ó┘ä┘ê╪» ╪¿┘ç ┘à╪º╪┤█î┘å╪ó┘ä╪º╪¬ ┘é╪º┘ä╪¿ ╪»╪º╪«┘ä█î ╪»╪º╪▒┘å╪» [12]. ╪º█î┘å ┘à┘ê╪º┘å╪╣ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿╪▒╪º█î ┘ê╪▒┘ê╪» ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å╪» ┘ê ╪▓┘à╪º┘å ╪º╪┤┌⌐╪º┘ä╪▓╪»╪º█î█î ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º┘ü╪▓╪º█î╪┤ ┘à█î╪»┘ç╪». **┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪»╪▒┘ê┘å┌»╪▒█î ┘à╪¡╪»┘ê╪»:** ┘é╪º┘ä╪¿┘ç╪º█î ╪│┘å╪¬█î ┘å┘à█î╪¬┘ê╪º┘å┘å╪» ┘à╪│╪¬┘é█î┘à╪º┘ï ╪│╪º╪«╪¬╪º╪▒ ╪º┘å┘ê╪º╪╣ ╪▒╪º ╪¿╪▒╪▒╪│█î ┌⌐┘å┘å╪». ╪¬┌⌐┘å█î┌⌐┘ç╪º█î█î ┘à╪º┘å┘å╪» SFINAE ┘ê std::enable\_if ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¡╪»┘ê╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪»╪î ╪º┘à╪º ╪º█î┘å┘ç╪º ╪»╪│╪¬┘ê┘╛╪º┌»█î╪▒ ┘ê ╪º╪║┘ä╪¿ ╪┤┌⌐┘å┘å╪»┘ç ┘ç╪│╪¬┘å╪» [13]. ┘ü┘é╪»╪º┘å ╪»╪▒┘ê┘å┌»╪▒█î ╪¼╪º┘à╪╣ ┘å┘ê╪╣ ┘à┘å╪¼╪▒ ╪¿┘ç ╪º┘å╪¬╪┤╪º╪▒ ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪«╪º╪▒╪¼█î ┘ê ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘à╪º┌⌐╪▒┘ê ╪┤╪»┘ç ╪º╪│╪¬. **┌å╪º┘ä╪┤┘ç╪º█î ┘å┌»┘ç╪»╪º╪▒█î:** ┌⌐╪» ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪»╪▒┌⌐╪î ╪¬╪║█î█î╪▒ ┘ê ┌»╪│╪¬╪▒╪┤ ╪ó┘å ╪»╪┤┘ê╪º╪▒ ╪º╪│╪¬. ╪╣╪»┘à ╪º╪▒╪¬╪¿╪º╪╖ ╪¿█î┘å ╪»╪º┘à┘å┘ç ┘à╪│╪ª┘ä┘ç ┘ê ╪▒╪º┘ç╪¡┘ä ┘é╪º┘ä╪¿ ╪º╪║┘ä╪¿ ┘à┘å╪¼╪▒ ╪¿┘ç ┌⌐╪»█î ┘à█î╪┤┘ê╪» ┌⌐┘ç ╪¿╪º┘ç┘ê╪┤ ╪º┘à╪º ╪║█î╪▒┘é╪º╪¿┘ä ┘å┌»┘ç╪»╪º╪▒█î ╪º╪│╪¬ [14]. ╪º█î┘å ╪º┘à╪▒ ╪¿┘ç ┘ê█î┌ÿ┘ç ╪»╪▒ ┘à╪¡█î╪╖┘ç╪º█î ╪│╪º╪▓┘à╪º┘å█î ┌⌐┘ç ┌⌐╪» ╪¿╪º█î╪» ╪¬┘ê╪│╪╖ ╪¬█î┘à┘ç╪º ╪¿╪▒╪º█î ╪»┘ê╪▒┘ç┘ç╪º█î ╪╖┘ê┘ä╪º┘å█î ┘å┌»┘ç╪»╪º╪▒█î ╪┤┘ê╪»╪î ┘à╪┤┌⌐┘ä╪│╪º╪▓ ╪º╪│╪¬. **╪¬┘ê╪▒┘à ╪¿╪º█î┘å╪▒█î:** ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ╪¿█î╪┤ ╪º╪▓ ╪¡╪» ┘é╪º┘ä╪¿ ┘à█î╪¬┘ê╪º┘å╪» ┘à┘å╪¼╪▒ ╪¿┘ç ╪º┘ü╪▓╪º█î╪┤ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪┤┘ê╪»╪î ╪¿┘ç ┘ê█î┌ÿ┘ç ╪▓┘à╪º┘å█î ┌⌐┘ç ╪¿╪º ╪»╪▒┘ê┘å╪«╪╖█î╪│╪º╪▓█î ╪¬┘ç╪º╪¼┘à█î ╪¬╪▒┌⌐█î╪¿ ╪┤┘ê╪» [15]. ╪º█î┘å ╪º┘à╪▒ ╪¿╪▒ ╪º┘å╪»╪º╪▓┘ç ╪º╪│╪¬┘é╪▒╪º╪▒╪î ╪▓┘à╪º┘å┘ç╪º█î ╪¿╪º╪▒┌»╪░╪º╪▒█î ┘ê ╪╣┘à┘ä┌⌐╪▒╪» ╪¡╪º┘ü╪╕┘ç ┘å┘ç╪º┘å ╪¬╪ú╪½█î╪▒ ┘à█î┌»╪░╪º╪▒╪». ### 1.3 ┘ê╪╣╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐╪î ┘ç┘à╪º┘å╪╖┘ê╪▒ ┌⌐┘ç ╪»╪▒ C++23 ┘à╪╣╪▒┘ü█î ╪┤╪»┘ç╪î ╪º█î┘å ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ╪▒╪º ╪¿╪º ╪º╪▒╪º╪ª┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à╪│╪¬┘é█î┘à ┘ê ╪»╪▒╪¼┘ç █î┌⌐ ╪▓╪¿╪º┘å ╪¿╪▒╪º█î ╪»╪▒┘ê┘å┌»╪▒█î ┘å┘ê╪╣ ┘ê ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒╪╖╪▒┘ü ┘à█î┌⌐┘å╪» [16╪î 17]. ╪¿╪▒╪«┘ä╪º┘ü ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┌⌐┘ç ╪»╪▒ ╪▓╪¿╪º┘å┘ç╪º█î█î ┘à╪º┘å┘å╪» Java █î╪º C# █î╪º┘ü╪¬ ┘à█î╪┤┘ê┘å╪»╪î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ C++23 ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¬┘ê┘ä█î╪» ┘ê ╪¬╪¡┘ä█î┘ä ┘é╪»╪▒╪¬┘à┘å╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪». ╪¿█î┘å╪┤ ┌⌐┘ä█î╪»█î ┘╛╪┤╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪º█î┘å ╪º╪│╪¬ ┌⌐┘ç ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪º╪▓ ┘é╪¿┘ä ╪º╪╖┘ä╪º╪╣╪º╪¬ ┌⌐╪º┘à┘ä█î ╪»╪▒╪¿╪º╪▒┘ç ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ╪»╪▒ ╪╖┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ ╪º╪«╪¬█î╪º╪▒ ╪»╪º╪▒╪». ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪│┘å╪¬█î ╪º╪│╪º╪│╪º┘ï ╪º█î┘å ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ┘à╪º╪┤█î┘å╪ó┘ä╪º╪¬ ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪¿╪º╪▓╪│╪º╪▓█î ┘à█î┌⌐┘å╪». ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪»╪▒ ╪╣┘ê╪╢ ╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à ╪¿┘ç ┘å┘à╪º█î╪┤ ╪»╪º╪«┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» ┘ê ┘å█î╪º╪▓ ╪¿┘ç ┌ÿ█î┘à┘å╪º╪│╪¬█î┌⌐ ┘é╪º┘ä╪¿ ╪▒╪º ╪¡╪░┘ü ┘à█î┌⌐┘å╪» [18]. ╪º█î┘å ╪▒┘ê█î┌⌐╪▒╪» ┌å┘å╪»█î┘å ┘à╪▓█î╪¬ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»: **╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à ╪¿┘ç ╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê╪╣:** ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪│╪¬╪▒╪│█î ┘ü┘ê╪▒█î ╪¿┘ç ┘å╪º┘à┘ç╪º█î ╪╣╪╢┘ê╪î ╪º┘å┘ê╪º╪╣╪î ┘ê█î┌ÿ┌»█î┘ç╪º ┘ê ╪▒┘ê╪º╪¿╪╖ ╪¿╪»┘ê┘å ╪º╪│╪¬┘å╪¬╪º╪¼ ┘╛█î┌å█î╪»┘ç ┘é╪º┘ä╪¿ ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪». **╪¿┘ç╪¿┘ê╪» ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪º┘à┘╛╪º█î┘ä:** ╪¿╪º ╪¡╪░┘ü ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿╪î ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪║┘ä╪¿ ╪│╪▒█î╪╣╪¬╪▒ ╪º╪▓ ┘à╪╣╪º╪»┘ä┘ç╪º█î ┘é╪º┘ä╪¿ ╪«┘ê╪» ┌⌐╪º┘à┘╛╪º█î┘ä ┘à█î╪┤┘ê┘å╪». **╪«┘ê╪º┘å╪º█î█î ╪¿┘ç╪¿┘ê╪» █î╪º┘ü╪¬┘ç:** ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪│╪¬┘é█î┘à╪º┘ï ┘é╪╡╪» ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│ ╪▒╪º ╪¿█î╪º┘å ┘à█î┌⌐┘å╪» ┘ê ╪ó┘å ╪▒╪º ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å█î ┌⌐┘ç ╪¿╪º ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘é╪º┘ä╪¿ ╪ó╪┤┘å╪º ┘å█î╪│╪¬┘å╪»╪î ┘é╪º╪¿┘ä ╪»╪│╪¬╪▒╪│╪¬╪▒ ┘à█î┌⌐┘å╪». **╪¬┘ê┘ä█î╪» ┌⌐╪» ┘é╪»╪▒╪¬┘à┘å╪»:** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘å╪º ┘à█î╪┤┘ê┘å╪» ╪¬╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪» ╪║█î╪▒╪╢╪▒┘ê╪▒█î╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪º┘ä┌»┘ê█î ╪╖╪▒╪º╪¡█î ┘ê ╪╣┘à┘ä┌⌐╪▒╪» ╪«╪º╪╡ ╪»╪º┘à┘å┘ç ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ╪│╪º╪▓┘å╪». ### 1.4 ╪º┘ç╪»╪º┘ü ╪¬╪¡┘é█î┘é ┘ê ┘à╪┤╪º╪▒┌⌐╪¬┘ç╪º ╪º█î┘å ┘à┘é╪º┘ä┘ç ┌å┘å╪»█î┘å ┘à╪┤╪º╪▒┌⌐╪¬ ┌⌐┘ä█î╪»█î ╪»╪▒ ╪»╪▒┌⌐ ┘ê ┌⌐╪º╪▒╪¿╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪»╪º╪▒╪»: **╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ┘ê█î┌ÿ┌»█î:** ┘à╪º ╪¬╪¡┘ä█î┘ä ┘ü┘å█î ╪»┘é█î┘é█î ╪º╪▓ API ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç█î┘à ┌⌐┘ç ┘é╪º╪¿┘ä█î╪¬┘ç╪º╪î ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ┘ê █î┌⌐┘╛╪º╪▒┌å┌»█î ╪ó┘å ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à┘ê╪¼┘ê╪» C++ ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». **╪º╪▒╪▓█î╪º╪¿█î ╪╣┘à┘ä┌⌐╪▒╪»:** ╪º╪▓ ╪╖╪▒█î┘é ╪º╪▒╪▓█î╪º╪¿█î ╪│█î╪│╪¬┘à╪º╪¬█î┌⌐ ╪╣┘à┘ä┌⌐╪▒╪» ╪»╪▒ ┌å┘å╪»█î┘å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒╪î ┘à╪º ╪¬╪ú╪½█î╪▒╪º╪¬ ╪╣┘à┘ä┌⌐╪▒╪»█î ╪▒┘ê╪┤┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ┘à┘é╪º╪¿┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪│┘å╪¬█î ╪▒╪º ┌⌐┘à█î╪│╪º╪▓█î ┘à█î┌⌐┘å█î┘à. **┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪╣┘à┘ä█î:** ┘à╪º ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç█î┘à ┌⌐┘ç ┌⌐╪º╪▒╪¿╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪¿╪▒╪º█î ┘à╪│╪º╪ª┘ä ╪▒╪º█î╪¼ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î╪î ╪º╪▓ ╪¼┘à┘ä┘ç ╪│╪▒█î╪º┘ä╪│╪º╪▓█î╪î ORM ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç╪î ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î GUI ┘ê ╪▓█î╪▒╪│╪º╪«╪¬ ╪¬╪│╪¬ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪». **┌å╪º╪▒┌å┘ê╪¿ ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º:** ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¡┘ä█î┘ä ┘ê ╪ó╪▓┘à╪º█î╪┤┘ç╪º█î ┘à╪º╪î ╪▒┘ç┘å┘à┘ê╪»┘ç╪º█î█î ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┘à╪ñ╪½╪▒ ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪»╪▒ ┌⌐╪» ╪¬┘ê┘ä█î╪»█î ┘╛█î╪┤┘å┘ç╪º╪» ┘à█î┌⌐┘å█î┘à. **╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç:** ┘à╪º ┘ü╪▒╪╡╪¬┘ç╪º█î█î ╪¿╪▒╪º█î ╪¿┘ç╪¿┘ê╪» ╪¿█î╪┤╪¬╪▒ ┘ê █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ ╪┤┘å╪º╪│╪º█î█î ┘à█î┌⌐┘å█î┘à ┘ê ┘å┘é╪┤┘ç ╪▒╪º┘ç█î ╪¿╪▒╪º█î ╪¬┌⌐╪º┘à┘ä ┘à╪»╪º┘ê┘à ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç█î┘à. ### 1.5 ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪¬╪¡┘é█î┘é ┘à╪º ╪¬╪¡┘ä█î┘ä ┘å╪╕╪▒█î ╪▒╪º ╪¿╪º ╪º╪▒╪▓█î╪º╪¿█î ╪¬╪¼╪▒╪¿█î ╪¬╪▒┌⌐█î╪¿ ┘à█î┌⌐┘å╪»: **╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º:** ╪¿╪▒╪▒╪│█î ╪»┘é█î┘é ╪º╪│┘å╪º╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ISO C++23 ┘ê ┘à┘é╪º┘ä╪º╪¬ ┘╛█î╪┤┘å┘ç╪º╪»█î ┘à╪▒╪¬╪¿╪╖ (P0194╪î P0385╪î P0707) [19╪î 20╪î 21]. **┘à╪╖╪º┘ä╪╣┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î:** ╪¬╪¡┘ä█î┘ä ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪º╪╡┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒╪î ╪º╪▓ ╪¼┘à┘ä┘ç ╪¿╪▒╪▒╪│█î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î. **╪º╪▒╪▓█î╪º╪¿█î ╪╣┘à┘ä┌⌐╪▒╪»:** ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪│█î╪│╪¬┘à╪º╪¬█î┌⌐ ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä╪î ╪º┘å╪»╪º╪▓┘ç┘ç╪º█î ╪¿╪º█î┘å╪▒█î ┘ê ╪╣┘à┘ä┌⌐╪▒╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¼┘à┘ê╪╣┘ç┘ç╪º█î ╪¬╪│╪¬ ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ╪»╪▒ ╪»╪º┘à┘å┘ç┘ç╪º█î ┘à╪«╪¬┘ä┘ü ┘à╪│╪ª┘ä┘ç. **╪¬┘ê╪│╪╣┘ç ┘à╪╖╪º┘ä╪╣┘ç ┘à┘ê╪▒╪»█î:** ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘å┘à╪º█î┘å╪»┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘ç╪▒ ╪»┘ê ╪▒┘ê╪┤ ╪│┘å╪¬█î ┘ê ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿╪î ╪¿╪º ╪¬╪¡┘ä█î┘ä ┘à┘é╪º█î╪│┘ç╪º█î ╪¬┘ä╪º╪┤ ╪¬┘ê╪│╪╣┘ç╪î ┘å┌»┘ç╪»╪º╪▒█î ┘ê ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪». **┘à╪╡╪º╪¡╪¿┘ç ╪¿╪º ╪«╪¿╪▒┌»╪º┘å:** ┘à╪┤┘ê╪▒╪¬ ╪¿╪º ╪º╪╣╪╢╪º█î ┌⌐┘à█î╪¬┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î C++╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ê ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪¿╪▒╪º█î ╪¼┘à╪╣╪ó┘ê╪▒█î ╪¿█î┘å╪┤┘ç╪º█î█î ╪»╪▒╪¿╪º╪▒┘ç ┘à┘ä╪º╪¡╪╕╪º╪¬ ╪╣┘à┘ä█î ┘ê ╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç. ### 1.6 ╪│╪º╪▓┘à╪º┘å╪»┘ç█î ┘à┘é╪º┘ä┘ç ╪¿╪º┘é█î ╪º█î┘å ┘à┘é╪º┘ä┘ç ╪¿┘ç ╪┤╪▒╪¡ ╪▓█î╪▒ ╪│╪º╪▓┘à╪º┘å╪»┘ç█î ╪┤╪»┘ç ╪º╪│╪¬: **╪¿╪«╪┤ 2** ┘╛█î╪┤╪▓┘à█î┘å┘ç ╪¬┌⌐╪º┘à┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ ┘ê ╪¿╪▒╪▒╪│█î ┌⌐╪º╪▒┘ç╪º█î ┘à╪▒╪¬╪¿╪╖ ╪»╪▒ ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪«╪¬┘ä┘ü ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». **╪¿╪«╪┤ 3** ╪¬╪¡┘ä█î┘ä ┘ü┘å█î ╪»┘é█î┘é█î ╪º╪▓ ┌å╪º╪▒┌å┘ê╪¿ ╪¿╪º╪▓╪¬╪º╪¿ C++23╪î ╪º╪▓ ╪¼┘à┘ä┘ç ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î API ┘ê ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». **╪¿╪«╪┤ 4** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪¿┘ç ╪╣┘à┘é ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪» ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î╪î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪▒╪º ┘╛┘ê╪┤╪┤ ┘à█î╪»┘ç╪». **╪¿╪«╪┤ 5** ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪╣┘à┘ä┌⌐╪▒╪» ┘ê ┘å╪¬╪º█î╪¼ ╪º╪▒╪▓█î╪º╪¿█î ╪╣┘à┘ä┌⌐╪▒╪» ┘à┘é╪º█î╪│┘ç ╪▒┘ê╪┤┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘é╪º┘ä╪¿ ╪│┘å╪¬█î ╪▒╪º ┌»╪▓╪º╪▒╪┤ ┘à█î╪»┘ç╪». **╪¿╪«╪┤ 6** ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ╪»┘é█î┘é█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» ┌⌐┘ç ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪╣┘à┘ä█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ┘å╪┤╪º┘å ┘à█î╪»┘ç╪». **╪¿╪«╪┤ 7** ┘ü╪▒╪╡╪¬┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++╪î ╪º╪▓ ╪¼┘à┘ä┘ç ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º╪î ┌⌐┘ê╪▒┘ê╪¬█î┘å┘ç╪º ┘ê ┘à╪º┌ÿ┘ê┘ä┘ç╪º ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». **╪¿╪«╪┤ 8** ┌å╪º┘ä╪┤┘ç╪º╪î ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ┘ê ╪¬┘ä┘ç┘ç╪º█î ╪º╪¡╪¬┘à╪º┘ä█î ╪»╪▒ ┘╛╪░█î╪▒╪┤ ╪▒┘ê╪┤┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¿╪¡╪½ ┘à█î┌⌐┘å╪». **╪¿╪«╪┤ 9** ╪¼┘ç╪¬┘ç╪º█î ╪¬╪¡┘é█î┘é ╪ó█î┘å╪»┘ç ┘ê ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ╪º╪¡╪¬┘à╪º┘ä█î ╪¿╪▒╪º█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î ╪¿╪╣╪»█î C++ ╪▒╪º ┘à╪┤╪«╪╡ ┘à█î┌⌐┘å╪». **╪¿╪«╪┤ 10** ╪¿╪º ╪«┘ä╪º╪╡┘ç╪º█î ╪º╪▓ █î╪º┘ü╪¬┘ç┘ç╪º█î ┌⌐┘ä█î╪»█î ┘ê ╪¬┘ê╪╡█î┘ç┘ç╪º█î█î ╪¿╪▒╪º█î ┘à╪¬╪«╪╡╪╡╪º┘å ┘å╪¬█î╪¼┘ç┌»█î╪▒█î ┘à█î┌⌐┘å╪». ╪º█î┘å ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪¿╪º ┘ç╪»┘ü ╪º╪▒╪º╪ª┘ç ┘ç┘à ╪¿█î┘å╪┤┘ç╪º█î ┘å╪╕╪▒█î ┘ê ┘ç┘à ╪▒╪º┘ç┘å┘à╪º█î█î ╪╣┘à┘ä█î ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å█î ┌⌐┘ç ╪¿┘ç ╪»┘å╪¿╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪»╪▒ ╪¬┘ä╪º╪┤┘ç╪º█î ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ╪«┘ê╪» ┘ç╪│╪¬┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 1-21 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à┘å╪º╪¿╪╣ ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 2. ┘╛█î╪┤╪▓┘à█î┘å┘ç ┘ê ┌⌐╪º╪▒┘ç╪º█î ┘à╪▒╪¬╪¿╪╖ ## 2.1 ╪¬┌⌐╪º┘à┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ (C++11 ╪¬╪º C++20) ### 2.1.1 ╪º┘å┘é┘ä╪º╪¿ ┘é╪º┘ä╪¿ (C++11) ┘à╪╣╪▒┘ü█î C++11 ┘å┘é╪╖┘ç ╪╣╪╖┘ü█î ╪»╪▒ ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ ╪¿┘ê╪». ┘é╪º┘ä╪¿┘ç╪º█î variadic ╪¿┘ç ╪╖┘ê╪▒ ╪¿┘å█î╪º╪»█î ┘å╪¡┘ê┘ç ┘å┘ê╪┤╪¬┘å ┌⌐╪» ╪╣┘à┘ê┘à█î ╪▒╪º ╪¬╪║█î█î╪▒ ╪»╪º╪»┘å╪» ┘ê ╪º█î╪¼╪º╪» ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î ┘é╪º┘ä╪¿ ┘ê╪º┘é╪╣╪º┘ï ╪º┘å╪╣╪╖╪º┘ü┘╛╪░█î╪▒ ┘ê ┘é╪º╪¿┘ä ╪º╪│╪¬┘ü╪º╪»┘ç ┘à╪¼╪»╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪»┘å╪» [22]. ┘ê█î┌ÿ┌»█î┘ç╪º█î ┌⌐┘ä█î╪»█î ╪▓█î╪▒ ┌å╪┤┘à╪º┘å╪»╪º╪▓ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▒╪º ╪¬╪║█î█î╪▒ ╪»╪º╪»┘å╪»: **┘é╪º┘ä╪¿┘ç╪º█î Variadic:** ┘é╪¿┘ä ╪º╪▓ C++11╪î ┘ü┘ç╪▒╪│╪¬ ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î ┘é╪º┘ä╪¿ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪½╪º╪¿╪¬ ╪¿┘ê╪»┘å╪» ┌⌐┘ç ┘à┘å╪¼╪▒ ╪¿┘ç ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪¡╪»┘ê╪» █î╪º ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┌»╪│╪¬╪▒╪»┘ç ╪¬╪«╪╡╪╡█î╪º╪¿█î ┘é╪º┘ä╪¿ ┘à█î╪┤╪». ┘é╪º┘ä╪¿┘ç╪º█î variadic ┌»╪│╪¬╪▒╪┤ ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿ ╪¿╪º ╪¬╪╣╪»╪º╪» ┘╛╪º╪▒╪º┘à╪¬╪▒ ╪»┘ä╪«┘ê╪º┘ç ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪»┘å╪»: cpp // ┘é╪¿┘ä ╪º╪▓ C++11: ┘à╪¡╪»┘ê╪» ╪¿┘ç ╪¬╪╣╪»╪º╪» ┘╛╪º╪▒╪º┘à╪¬╪▒ ╪½╪º╪¿╪¬ template<typename T1, typename T2, typename T3> struct tuple3; template<typename T1, typename T2> struct tuple2; // ... ╪¬╪«╪╡╪╡█î╪º╪¿█î┘ç╪º█î ┌»╪│╪¬╪▒╪»┘ç // C++11: ╪¬╪╣╪»╪º╪» ┘╛╪º╪▒╪º┘à╪¬╪▒ ╪»┘ä╪«┘ê╪º┘ç template<typename... Types> struct tuple; **Perfect Forwarding:** ╪¬╪▒┌⌐█î╪¿ rvalue references ┘ê universal references (T&&) perfect forwarding ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪» ┘ê ╪¿┘ç ╪¬┘ê╪º╪¿╪╣ ┘é╪º┘ä╪¿ ╪º╪¼╪º╪▓┘ç ╪»╪º╪» ╪»╪│╪¬┘ç ┘à┘é╪»╪º╪▒ ╪ó╪▒┌»┘ê┘à╪º┘å┘ç╪º█î ╪«┘ê╪» ╪▒╪º ╪¡┘ü╪╕ ┌⌐┘å┘å╪» [23]. ╪º█î┘å ┘╛█î╪┤╪▒┘ü╪¬ ┘à╪│╪º╪ª┘ä ╪╖┘ê┘ä╪º┘å█î┘à╪»╪¬ ╪»╪▒ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ┘à╪▒╪¬╪¿╪╖ ╪¿╪º ┌⌐╪º╪▒╪º█î█î ┘ê ╪╡╪¡╪¬ ╪▒╪º ╪¡┘ä ┌⌐╪▒╪». **╪¿┘ç╪¿┘ê╪»┘ç╪º█î SFINAE:** Substitution Failure Is Not An Error (SFINAE) ╪¿╪º std::enable\_if ┘ê type trait┘ç╪º█î ┘à╪▒╪¬╪¿╪╖ ┘é╪»╪▒╪¬┘à┘å╪»╪¬╪▒ ┘ê ┘é╪º╪¿┘ä ╪º╪│╪¬┘ü╪º╪»┘ç╪¬╪▒ ╪┤╪» ┘ê resolution overload ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘å┘ê╪╣ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪» [24]. **╪¬┘ê╪º╪¿╪╣ Constexpr:** ┘à╪╣╪▒┘ü█î constexpr ╪¿┘ç ╪¿╪▒╪«█î ╪¬┘ê╪º╪¿╪╣ ╪º╪¼╪º╪▓┘ç ╪»╪º╪» ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪▒╪▓█î╪º╪¿█î ╪┤┘ê┘å╪» ┘ê ╪┤┌⌐╪º┘ü ╪¿█î┘å ┘à╪¡╪º╪│╪¿┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ê ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ┘╛╪▒ ┌⌐╪▒╪»┘å╪» [25]. ### 2.1.2 ╪¬╪│┘ä╪╖ Type Traits ┘ê SFINAE (C++14) C++14 ┘à╪»┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▒╪º ╪¿╪º ┌å┘å╪»█î┘å ╪¿┘ç╪¿┘ê╪» ┌⌐┘ä█î╪»█î ┌⌐┘ç ┌⌐╪» ┘é╪º┘ä╪¿ ╪▒╪º ╪«┘ê╪º┘å╪º╪¬╪▒ ┘ê ┘é╪º╪¿┘ä ┘å┌»┘ç╪»╪º╪▒█î╪¬╪▒ ┌⌐╪▒╪»╪î ╪¬╪╡┘ü█î┘ç ┌⌐╪▒╪» [26]: **┘é╪º┘ä╪¿┘ç╪º█î ┘à╪¬╪║█î╪▒:** ╪¬┘ê╪º┘å╪º█î█î ╪º█î╪¼╪º╪» ┘à╪¬╪║█î╪▒┘ç╪º█î ┘é╪º┘ä╪¿█î ╪¿╪│█î╪º╪▒█î ╪º╪▓ ┘à┘ê╪º╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç type trait ╪▒╪º ╪│╪º╪»┘ç ┌⌐╪▒╪»: cpp // C++11: ┘å╪¡┘ê ┘╛╪▒╪¡╪▒┘ü template<typename T> struct is\_integral : std::integral\_constant<bool, /\* ┘à┘å╪╖┘é ┘╛█î┌å█î╪»┘ç \*/> {}; // C++14: ┘é╪º┘ä╪¿ ┘à╪¬╪║█î╪▒ ╪│╪º╪»┘ç╪┤╪»┘ç template<typename T> constexpr bool is\_integral\_v = is\_integral<T>::value; **┘ä╪º┘à╪¿╪»╪º┘ç╪º█î ╪╣┘à┘ê┘à█î:** ┘ä╪º┘à╪¿╪»╪º┘ç╪º ╪¬┘ê╪º┘å╪º█î█î ┘╛╪░█î╪▒╪┤ ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î auto ╪▒╪º ╪¿┘ç ╪»╪│╪¬ ╪ó┘ê╪▒╪»┘å╪» ┘ê ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│╪¿┌⌐ ╪¬╪º╪¿╪╣█î ┌⌐┘ç ┘é╪¿┘ä╪º┘ï ╪║█î╪▒┘à┘à┌⌐┘å █î╪º ╪»╪│╪¬┘ê┘╛╪º┌»█î╪▒ ╪¿┘ê╪»┘å╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪»┘å╪» [27]. **Constexpr ┌»╪│╪¬╪▒╪»┘ç:** ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪ó╪│╪º┘å╪¬╪▒ ╪┤╪»┘ç ╪▒┘ê█î ╪¬┘ê╪º╪¿╪╣ constexpr ┘à╪¡╪º╪│╪¿╪º╪¬ ┘╛█î┌å█î╪»┘ç╪¬╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪î ╪º╪▓ ╪¼┘à┘ä┘ç ╪¡┘ä┘é┘ç┘ç╪º ┘ê ╪╣╪¿╪º╪▒╪º╪¬ ╪┤╪▒╪╖█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┌⌐╪▒╪» [28]. ### 2.1.3 ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ┘ê ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º (C++20) C++20 ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ╪▒╪º ┘à╪╣╪▒┘ü█î ┌⌐╪▒╪» ┌⌐┘ç ┘à┘ç┘à╪¬╪▒█î┘å ┘╛█î╪┤╪▒┘ü╪¬ ╪»╪▒ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î C++ ╪º╪▓ ╪▓┘à╪º┘å ┘é╪º┘ä╪¿┘ç╪º█î variadic ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [29]. ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ┌å┘å╪»█î┘å ┘à╪│╪ª┘ä┘ç ╪¿┘å█î╪º╪»█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ╪▒╪º ╪¡┘ä ┘à█î┌⌐┘å┘å╪»: **╪╣╪¿╪º╪▒╪¬ ┘à╪¡╪»┘ê╪»█î╪¬ ┘é╪º┘ä╪¿:** ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ╪▒╪º┘ç█î ╪º╪╣┘ä╪º┘å█î ╪¿╪▒╪º█î ╪¬╪╣█î█î┘å ╪º┘ä╪▓╪º┘à╪º╪¬ ┘é╪º┘ä╪¿ ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å┘å╪» ┘ê ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┘╛█î┌å█î╪»┘ç SFINAE ╪▒╪º ╪¿╪º ╪╣╪¿╪º╪▒╪º╪¬ ┘à╪¡╪»┘ê╪»█î╪¬ ╪«┘ê╪º┘å╪º ╪¼╪º█î┌»╪▓█î┘å ┘à█î┌⌐┘å┘å╪»: cpp // ╪▒┘ê█î┌⌐╪▒╪» ╪│┘å╪¬█î SFINAE template<typename T> typename std::enable\_if\_t< std::is\_arithmetic\_v<T> && std::is\_copy\_constructible\_v<T> && requires(T a, T b) { a + b; }, T > add(T a, T b) { return a + b; } // ╪▒┘ê█î┌⌐╪▒╪» ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º template<typename T> concept Addable = std::is\_arithmetic\_v<T> && std::is\_copy\_constructible\_v<T> && requires(T a, T b) { a + b; }; template<Addable T> T add(T a, T b) { return a + b; } **╪¿┘ç╪¿┘ê╪» ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º:** ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ╪¬╪┤╪«█î╪╡ ╪«╪╖╪º█î ┘é╪º┘ä╪¿ ╪▒╪º ╪¿╪º ╪º╪▒╪º╪ª┘ç ┘╛█î╪º┘à┘ç╪º█î ┘ê╪º╪╢╪¡ ┘å┘é╪╢ ┘à╪¡╪»┘ê╪»█î╪¬ ╪¿┘ç ╪¼╪º█î ╪«╪╖╪º┘ç╪º█î ╪╣┘à█î┘é ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ┘é╪º┘ä╪¿ ╪¿┘ç ╪╖┘ê╪▒ ┌å╪┤┘à┌»█î╪▒█î ╪¿┘ç╪¿┘ê╪» ┘à█î╪¿╪«╪┤┘å╪» [30]. **╪¿┘ç╪¿┘ê╪» Resolution Overload:** ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º resolution overload ╪»┘é█î┘é╪¬╪▒ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å┘å╪» ┘ê ╪¿┘ç ┘å┘ê█î╪│┘å╪»┌»╪º┘å ┘é╪º┘ä╪¿ ╪º╪¼╪º╪▓┘ç ┘à█î╪»┘ç┘å╪» ╪«╪º┘å┘ê╪º╪»┘ç┘ç╪º█î█î ╪º╪▓ ╪¬┘ê╪º╪¿╪╣ ┘à╪▒╪¬╪¿╪╖ ╪¿╪º ┘é┘ê╪º┘å█î┘å ╪º┘ê┘ä┘ê█î╪¬ ┘ê╪º╪╢╪¡ ╪º█î╪¼╪º╪» ┌⌐┘å┘å╪» [31]. ### 2.1.4 ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪╣┘ä█î╪▒╪║┘à ╪º█î┘å ┘╛█î╪┤╪▒┘ü╪¬┘ç╪º╪î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪¿┘å█î╪º╪»█î ╪▒╪º ╪¡┘ü╪╕ ┌⌐╪▒╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ç╪»┘ü ╪▒┘ü╪╣ ╪ó┘å┘ç╪º ╪▒╪º ╪»╪º╪▒╪» [32]: **╪»╪▒┘ê┘å┌»╪▒█î ╪║█î╪▒┘à╪│╪¬┘é█î┘à ┘å┘ê╪╣:** ┘é╪º┘ä╪¿┘ç╪º ╪¬┘å┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º┘å┘ê╪º╪╣ ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪▒╪º╪¿╪╖ ╪ó┘å┘ç╪º╪î ┘å┘ç ╪│╪º╪«╪¬╪º╪▒ ╪»╪º╪«┘ä█î╪┤╪º┘å╪î ╪¿╪▒╪▒╪│█î ┌⌐┘å┘å╪». ╪º█î┘å ┘à╪¡╪»┘ê╪»█î╪¬ ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘╛█î┌å█î╪»┘ç╪º█î ╪¿╪▒╪º█î ┌⌐╪º╪▒┘ç╪º█î█î ┘à╪º┘å┘å╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ █î╪º ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪º█î╪¼╪º╪¿ ┘à█î┌⌐┘å╪». **┘╛█î┌å█î╪»┌»█î ┌⌐╪º┘à┘╛╪º█î┘ä:** ┘ç┘à╪º┘å╪╖┘ê╪▒ ┌⌐┘ç ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘é╪º┘ä╪¿ ╪╣┘à█î┘é╪¬╪▒ ┘ê ┘╛█î┌å█î╪»┘ç╪¬╪▒ ┘à█î╪┤┘ê┘å╪»╪î ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ┘à█î╪¬┘ê╪º┘å┘å╪» ┘à┘à┘å┘ê╪╣┌⌐┘å┘å╪»┘ç ╪┤┘ê┘å╪». ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┌⌐┘ç ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ┘╛█î┌å█î╪»┘ç ┘é╪º┘ä╪¿ ┘à█î╪¬┘ê╪º┘å╪» 60-80% ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ ┘╛╪º█î┌»╪º┘ç┘ç╪º█î ┌⌐╪» ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ ╪▒╪º ╪¬╪┤┌⌐█î┘ä ╪»┘ç╪» [33]. **╪¿╪º╪▒ ╪┤┘å╪º╪«╪¬█î:** ╪┤┌⌐╪º┘ü ╪¿█î┘å ╪¿█î╪º┘å ┘à╪│╪ª┘ä┘ç ┘ê ╪▒╪º┘ç╪¡┘ä ┘é╪º┘ä╪¿ ╪º╪║┘ä╪¿ ┘å█î╪º╪▓ ╪¿┘ç ╪¬╪«╪╡╪╡ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿╪▒╪º█î ┘╛┘ä ╪▓╪»┘å ╪»╪º╪▒╪». ╪º█î┘å ┘à┘ê╪º┘å╪╣█î ╪¿╪▒╪º█î ┘╛╪░█î╪▒╪┤ ┘ê ┌å╪º┘ä╪┤┘ç╪º█î ┘å┌»┘ç╪»╪º╪▒█î ╪»╪▒ ╪¬█î┘à┘ç╪º█î ╪¿╪▓╪▒┌» ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å╪» [34]. ## 2.2 ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪▓╪¿╪º┘å┘ç╪º█î ╪»█î┌»╪▒ ╪»╪▒┌⌐ ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘å█î╪º╪▓ ╪¿┘ç ╪¿╪▒╪▒╪│█î ┘å╪¡┘ê┘ç ╪▒┘ê█î╪º╪▒┘ê█î█î ╪│╪º█î╪▒ ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¿╪º ┘à╪│╪ª┘ä┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪º╪▒╪»╪î ╪▓█î╪▒╪º ╪º█î┘å ╪▒╪º┘ç╪¡┘ä┘ç╪º ╪¬╪╡┘à█î┘à╪º╪¬ ╪╖╪▒╪º╪¡█î C++ ╪▒╪º ╪ó┌»╪º┘ç ┌⌐╪▒╪»┘ç╪º┘å╪» [35]. ### 2.2.1 ╪¿╪º╪▓╪¬╪º╪¿ ╪¼╪º┘ê╪º (java.lang.reflect) ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪¼╪º┘ê╪º ┌⌐┘ç ╪»╪▒ Java 1.1 ┘à╪╣╪▒┘ü█î ╪┤╪»╪î ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¼╪º┘à╪╣ ╪»╪▒┘ê┘å┌»╪▒█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» [36]. ╪▒┘ê█î┌⌐╪▒╪» ╪¼╪º┘ê╪º ┌å┘å╪»█î┘å ╪¬╪╢╪º╪» ╪ó┘à┘ê╪▓┘å╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ C++ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»: **╪º┘å╪╣╪╖╪º┘ü┘╛╪░█î╪▒█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º:** ╪¿╪º╪▓╪¬╪º╪¿ ╪¼╪º┘ê╪º ╪»╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪╣┘à┘ä ┘à█î┌⌐┘å╪» ┘ê ╪¿╪º╪▒┌»╪░╪º╪▒█î ┌⌐┘ä╪º╪│ ┘╛┘ê█î╪º╪î ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪¬╪» ┘ê ╪»╪│╪¬╪▒╪│█î ┘ü█î┘ä╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪»: java Class<?> clazz = Class.forName("com.example.Person"); Method[] methods = clazz.getDeclaredMethods(); Field[] fields = clazz.getDeclaredFields(); Object instance = clazz.getDeclaredConstructor().newInstance(); Method setter = clazz.getMethod("setName", String.class); setter.invoke(instance, "Alice"); **╪│╪▒╪¿╪º╪▒ ╪╣┘à┘ä┌⌐╪▒╪»:** ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ç╪▓█î┘å┘ç┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿┘ç ╪»┘ä█î┘ä dispatch ┘╛┘ê█î╪º╪î ╪¿╪▒╪▒╪│█î┘ç╪º█î ╪º┘à┘å█î╪¬█î ┘ê ┘ü┘é╪»╪º┘å ┘ü╪▒╪╡╪¬┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┘à╪¬╪¡┘à┘ä ┘à█î╪┤┘ê╪» [37]. ┘à█î┌⌐╪▒┘ê╪º╪▒╪▓█î╪º╪¿█î┘ç╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┘ü╪▒╪º╪«┘ê╪º┘å█î┘ç╪º█î ┘à╪¬╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å┘å╪» 10-100 ╪¿╪▒╪º╪¿╪▒ ┌⌐┘å╪»╪¬╪▒ ╪º╪▓ ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪│╪¬┘é█î┘à ╪¿╪º╪┤┘å╪». **╪¬╪ú╪½█î╪▒╪º╪¬ ╪º┘à┘å█î╪¬█î:** ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪¼╪º┘ê╪º ┘å█î╪º╪▓ ╪¿┘ç ┘à╪»█î╪▒█î╪¬ ╪º┘à┘å█î╪¬█î ╪»┘é█î┘é ╪¿╪▒╪º█î ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪»╪│╪¬╪▒╪│█î ╪║█î╪▒┘à╪¼╪º╪▓ ╪¿┘ç ╪º╪╣╪╢╪º█î ╪«╪╡┘ê╪╡█î ┘ê ┘à┘å╪º╪¿╪╣ ╪│█î╪│╪¬┘à ╪»╪º╪▒╪» [38]. ### 2.2.2 ╪¿╪º╪▓╪¬╪º╪¿ C# (System.Reflection) ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ C# ╪¿╪▒ ┘╛╪º█î┘ç ╪¼╪º┘ê╪º ╪¿┘å╪º ┘à█î╪┤┘ê╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪▒╪º ╪º╪╢╪º┘ü┘ç ┘à█î┌⌐┘å╪» [39]: **┘ê█î┌ÿ┌»█î┘ç╪º ┘ê ┘ü╪▒╪º╪»╪º╪»┘ç:** C# ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¿╪º ╪│█î╪│╪¬┘à ┘ê█î┌ÿ┌»█î ╪║┘å█î █î┌⌐┘╛╪º╪▒┌å┘ç ┘à█î┌⌐┘å╪» ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º╪╣┘ä╪º┘å█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪»: csharp [Serializable] public class Person { [JsonProperty("full\_name")] public string Name { get; set; } [JsonIgnore] public int InternalId { get; set; } } // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ Type type = typeof(Person); PropertyInfo[] properties = type.GetProperties(); foreach (var prop in properties) { var jsonAttr = prop.GetCustomAttribute<JsonPropertyAttribute>(); // ┘╛╪▒╪»╪º╪▓╪┤ ╪¿╪▒ ╪º╪│╪º╪│ ┘ê█î┌ÿ┌»█î┘ç╪º } **╪»╪▒╪«╪¬┘ç╪º█î ╪╣╪¿╪º╪▒╪¬:** C# ╪»╪▒╪«╪¬┘ç╪º█î ╪╣╪¿╪º╪▒╪¬ ╪▒╪º ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘å┘à╪º█î╪┤ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┌⌐╪» ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» ┘ê ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î█î ┘à╪º┘å┘å╪» Entity Framework ╪▒╪º ┘é╪º╪»╪▒ ┘à█î╪│╪º╪▓╪» ╪╣╪¿╪º╪▒╪º╪¬ C# ╪▒╪º ╪¿┘ç ┌⌐┘ê╪ª╪▒█î┘ç╪º█î SQL ╪¬╪▒╪¼┘à┘ç ┌⌐┘å┘å╪» [40]. **╪¬┘ê┘ä█î╪»┌⌐┘å┘å╪»┌»╪º┘å ┘à┘å╪¿╪╣:** ┘å╪│╪«┘ç┘ç╪º█î ╪º╪«█î╪▒ C# ╪¬┘ê┘ä█î╪»┌⌐┘å┘å╪»┌»╪º┘å ┘à┘å╪¿╪╣ ╪▒╪º ┘à╪╣╪▒┘ü█î ┌⌐╪▒╪»┘å╪» ┌⌐┘ç ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘à╪┤╪º╪¿┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å┘å╪» [41]. ### 2.2.3 ┘à╪º┌⌐╪▒┘ê┘ç╪º█î ╪▒┘ê█î┘ç╪º█î Rust Rust ╪▒┘ê█î┌⌐╪▒╪» ┘à┘å╪¡╪╡╪▒╪¿┘ç┘ü╪▒╪»█î ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪▓ ╪╖╪▒█î┘é ┘à╪º┌⌐╪▒┘ê┘ç╪º█î ╪▒┘ê█î┘ç╪º█î ╪º╪¬╪«╪º╪░ ┘à█î┌⌐┘å╪» ┌⌐┘ç ╪▒┘ê█î ╪»╪▒╪«╪¬ ┘å╪¡┘ê ╪º┘å╪¬╪▓╪º╪╣█î (AST) ╪»╪▒ ╪╖┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪╣┘à┘ä ┘à█î┌⌐┘å┘å╪» [42]: **┌»╪│╪¬╪▒╪┤ ┘å╪¡┘ê:** ┘à╪º┌⌐╪▒┘ê┘ç╪º█î ╪▒┘ê█î┘ç╪º█î ┘à█î╪¬┘ê╪º┘å┘å╪» ┌⌐╪» Rust ╪»┘ä╪«┘ê╪º┘ç ╪¿╪▒ ╪º╪│╪º╪│ ┘å╪¡┘ê ┘ê╪▒┘ê╪»█î ╪¬┘ê┘ä█î╪» ┌⌐┘å┘å╪»: rust #[derive(Serialize, Debug)] struct Person { name: String, age: u32, } // ┘à╪º┌⌐╪▒┘ê derive ┌⌐╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: impl Serialize for Person { fn serialize(&self) -> String { // ┘à┘å╪╖┘é ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç } } **╪º╪¼╪▒╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä:** ┘à╪º┌⌐╪▒┘ê┘ç╪º█î Rust ╪»╪▒ ╪╖┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪¼╪▒╪º ┘à█î╪┤┘ê┘å╪» ┘ê ╪º┘å╪¬╪▓╪º╪╣╪º╪¬ ╪¿╪»┘ê┘å ┘ç╪▓█î┘å┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º ╪»╪▒ ╪¡█î┘å ╪¡┘ü╪╕ ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓┘å╪» [43]. **╪¿┘ç╪»╪º╪┤╪¬ ┘ê ╪º█î┘à┘å█î:** ╪│█î╪│╪¬┘à ┘à╪º┌⌐╪▒┘ê Rust ╪¬╪╢┘à█î┘å┘ç╪º█î ╪¿┘ç╪»╪º╪┤╪¬█î ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» ┘ê ╪º╪▓ capture ┘å╪º┘à ╪¬╪╡╪º╪»┘ü█î ╪¼┘ä┘ê┌»█î╪▒█î ┘à█î┌⌐┘å╪» ┘ê ╪▒┘ü╪¬╪º╪▒ ┘é╪º╪¿┘ä ┘╛█î╪┤╪¿█î┘å█î ╪¬╪╢┘à█î┘å ┘à█î┌⌐┘å╪» [44]. ### 2.2.4 ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▓╪¿╪º┘å D ╪▓╪¿╪º┘å ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î D ╪¿╪│█î╪º╪▒█î ╪º╪▓ ┘à┘ü╪º┘ç█î┘à█î ╪▒╪º ┘╛█î╪┤┌»╪º┘à█î ┌⌐╪▒╪» ┌⌐┘ç ╪¿╪▒ ╪╖╪▒╪º╪¡█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¬╪ú╪½█î╪▒ ┘à█î┌»╪░╪º╪▒┘å╪» [45]: **█î┌⌐┘╛╪º╪▒┌å┌»█î ┘é╪º┘ä╪¿ ┘ê Mixin:** D ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¿╪º ┘é╪º┘ä╪¿┘ç╪º ┘ê string mixin┘ç╪º ╪¿┘ç ╪╖┘ê╪▒ █î┌⌐┘╛╪º╪▒┌å┘ç █î┌⌐┘╛╪º╪▒┌å┘ç ┘à█î┌⌐┘å╪»: d struct Person { string name; int age; } // ╪¬┌⌐╪▒╪º╪▒ ┘ü█î┘ä╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä foreach (i, field; Person.tupleof) { writeln("Field ", i, ": ", typeof(field).stringof); } // String mixin ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» mixin(generateToString!Person); **╪»╪▒┘ê┘å┌»╪▒█î ╪º╪│╪¬╪º╪¬█î┌⌐:** D ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪¼╪º┘à╪╣ ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪»┘ê┘å ╪│╪▒╪¿╪º╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» [46]. **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪¬┘ê┘ä█î╪» ┌⌐╪»:** ╪¬╪▒┌⌐█î╪¿ ╪º╪¼╪▒╪º█î ╪¬╪º╪¿╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä (CTFE) ┘ê string mixin┘ç╪º ╪º┘ä┌»┘ê┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» [47]. ## 2.3 ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î ┘é╪¿┘ä█î ╪¿╪º╪▓╪¬╪º╪¿ C++ ┘à╪│█î╪▒ ╪¿┘ç ╪│┘à╪¬ ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪┤╪º┘à┘ä ╪¬┘ä╪º╪┤┘ç╪º█î ┌»╪│╪¬╪▒╪»┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ╪»╪▒ ┌å┘å╪»█î┘å ╪│╪º┘ä ┘ê ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î ┘à╪¬╪╣╪»╪» ╪¿┘ê╪» [48]. ### 2.3.1 P0194: ┘╛█î╪┤┘å┘ç╪º╪» ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ P0194 ┌⌐┘ç ╪¬┘ê╪│╪╖ Mat├║┼í Chochl├¡k╪î Axel Naumann ┘ê David Sankel ┘å┘ê╪┤╪¬┘ç ╪┤╪»╪î ┌⌐╪º╪▒ ╪¿┘å█î╪º╪»█î ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [49]. ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ┌⌐┘ä█î╪»█î ╪┤╪º┘à┘ä: **╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º:** ╪¬┘à╪º┘à ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪« ┘à█î╪»┘ç┘å╪» ┘ê ┘ç█î┌å ╪¼╪▒█î┘à┘ç ╪╣┘à┘ä┌⌐╪▒╪»█î ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬╪╢┘à█î┘å ┘å┘à█î┌⌐┘å┘å╪». **╪º█î┘à┘å█î ┘å┘ê╪╣:** ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪╖┘ê╪▒ ┌⌐╪º┘à┘ä ┘å┘ê╪╣-╪¿╪▒╪▒╪│█î ┘à█î╪┤┘ê┘å╪» ┘ê ╪º╪▓ ╪«╪╖╪º┘ç╪º█î ╪▒╪º█î╪¼ ┘à╪▒╪¬╪¿╪╖ ╪¿╪º ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪▒╪┤╪¬┘ç ╪¼┘ä┘ê┌»█î╪▒█î ┘à█î┌⌐┘å┘å╪». **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à┘ê╪¼┘ê╪»:** ┘╛█î╪┤┘å┘ç╪º╪» ╪¿╪º ╪»┘é╪¬ ╪¬╪╣╪º┘à┘ä ╪¿╪º ┘é╪º┘ä╪¿┘ç╪º╪î ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ┘ê ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î C++ ╪▒╪º ╪»╪▒ ┘å╪╕╪▒ ┘à█î┌»█î╪▒╪». **┘╛╪░█î╪▒╪┤ ╪¬╪»╪▒█î╪¼█î:** ╪╖╪▒╪º╪¡█î ╪¿┘ç ┌⌐╪» ┘à┘ê╪¼┘ê╪» ╪º╪¼╪º╪▓┘ç ┘à█î╪»┘ç╪» ╪¿╪»┘ê┘å ╪¬╪║█î█î╪▒ ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç╪▒┘ç┘à┘å╪» ╪┤┘ê╪». ### 2.3.2 P0385: ┘à┘å╪╖┘é ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪º█î┘å ┘╛█î╪┤┘å┘ç╪º╪» ┘à┘å╪╖┘é ╪»┘é█î┘é█î ╪¿╪▒╪º█î ╪¬╪╡┘à█î┘à╪º╪¬ ╪╖╪▒╪º╪¡█î ╪»╪▒ ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪▒╪º╪ª┘ç ╪»╪º╪» [50]: **┘ü┘ä╪│┘ü┘ç ╪╖╪▒╪º╪¡█î API:** API ╪¿╪º╪▓╪¬╪º╪¿ ┘é╪º╪¿┘ä█î╪¬ ╪¿█î╪º┘å ┘ê ╪º█î┘à┘å█î ╪▒╪º ╪¿╪▒ ╪º╪«╪¬╪╡╪º╪▒ ╪º┘ê┘ä┘ê█î╪¬ ┘à█î╪»┘ç╪» ┌⌐┘ç ┘à┘å╪¼╪▒ ╪¿┘ç ┌⌐╪» ┘╛╪▒╪¡╪▒┘ü╪¬╪▒ ╪º┘à╪º ╪º█î┘à┘å╪¬╪▒ ┘à█î╪┤┘ê╪». **┘à┘ä╪º╪¡╪╕╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î:** ┘╛█î╪┤┘å┘ç╪º╪» ┌å╪º┘ä╪┤┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪▒╪º ┘╛╪▒╪»╪º╪«╪¬┘ç ┘ê ╪▒╪º┘ç┘å┘à╪º█î█î ╪¿╪▒╪º█î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º╪▒╪ó┘à╪» ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪». **╪¬╪¡┘ä█î┘ä ┘à┘ê╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç:** ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ┘à┘ê╪º╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ╪¬╪╡┘à█î┘à╪º╪¬ ╪╖╪▒╪º╪¡█î API ╪▒╪º ╪ó┌»╪º┘ç ┌⌐╪▒╪». ### 2.3.3 P0707: ┘╛█î╪┤┘å┘ç╪º╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘╛█î╪┤┘å┘ç╪º╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î Herb Sutter ┘à┘ü┘ç┘ê┘à ╪¬┘ê┘ä█î╪» ┌⌐┘ä╪º╪│ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ┘à╪╣╪▒┘ü█î ┌⌐╪▒╪» [51]: **╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î:** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪▒╪º█î╪¼ ┘à╪º┘å┘å╪» ╪«╪╡┘ê╪╡█î╪º╪¬╪î ┘å╪╕╪º╪▒╪¬┌»╪▒╪º┘å ┘ê ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓┘å╪». **╪╖╪▒╪º╪¡█î ┘å╪¡┘ê:** ┘╛█î╪┤┘å┘ç╪º╪» ╪¿╪º ╪»┘é╪¬ ┘é╪º╪¿┘ä█î╪¬ ╪¿█î╪º┘å ╪▒╪º ╪¿╪º ┘ê╪╢┘ê╪¡ ┘å╪¡┘ê█î ┘à╪¬╪╣╪º╪»┘ä ┘à█î┌⌐┘å╪» ┘ê ╪º╪▓ ┘å╪¡┘ê ╪¿█î╪┤ ╪º╪▓ ╪¡╪» ┘╛█î┌å█î╪»┘ç █î╪º ╪▒┘à╪▓╪ó┘ä┘ê╪» ╪º╪¼╪¬┘å╪º╪¿ ┘à█î┌⌐┘å╪». **╪º╪│╪¬╪▒╪º╪¬┌ÿ█î █î┌⌐┘╛╪º╪▒┌å┌»█î:** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒ ╪º┘ê┘ä█î┘ç┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘å╪º ┘à█î╪┤┘ê┘å╪» ┘ê ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┌å┌»┘ê┘å┘ç ╪»┘ê ┘ê█î┌ÿ┌»█î ╪¿┘ç ╪╖┘ê╪▒ ┘ç┘à╪º┘ü╪▓╪º ╪¿╪º ┘ç┘à ┌⌐╪º╪▒ ┘à█î┌⌐┘å┘å╪». ### 2.3.4 ╪¬┌⌐╪º┘à┘ä ╪º╪▓ ╪╖╪▒█î┘é ┘ü╪▒╪ó█î┘å╪» ┌⌐┘à█î╪¬┘ç ┘ü╪▒╪ó█î┘å╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ╪┤╪º┘à┘ä ╪¬╪╡┘ü█î┘ç ┌»╪│╪¬╪▒╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¼╪▒╪¿┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ╪¿╪º╪▓╪«┘ê╪▒╪» ╪¼╪º┘à╪╣┘ç ╪¿┘ê╪» [52]: **╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î:** ┌å┘å╪»█î┘å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪ó╪▓┘à╪º█î╪┤█î ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒█î ┘ê ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ╪╖╪▒╪º╪¡█î┘ç╪º█î ┘╛█î╪┤┘å┘ç╪º╪»█î ╪▒╪º ╪º╪╣╪¬╪¿╪º╪▒ ╪»╪º╪»┘å╪». **╪¬╪╡┘ü█î┘ç API:** ┘ü╪▒╪ó█î┘å╪» ┌⌐┘à█î╪¬┘ç ┘à┘å╪¼╪▒ ╪¿┘ç ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘à╪¬╪╣╪»╪» API ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ┘ê ╪¿╪º╪▓╪«┘ê╪▒╪» ╪º╪▓ ┘╛╪░█î╪▒┘å╪»┌»╪º┘å ╪º┘ê┘ä█î┘ç ╪┤╪». **╪¬╪│╪¬ █î┌⌐┘╛╪º╪▒┌å┌»█î:** ╪¬╪│╪¬ ┌»╪│╪¬╪▒╪»┘ç █î┌⌐┘╛╪º╪▒┌å┌»█î ┘à┘å╪º╪│╪¿ ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º ┘ê ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î ┘à┘ê╪¼┘ê╪» C++ ╪▒╪º ╪¬╪╢┘à█î┘å ┌⌐╪▒╪». ## 2.4 ╪¬╪¡┘ä█î┘ä ┘à┘é╪º█î╪│┘ç╪º█î ### 2.4.1 ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪»╪▒ ┘à┘é╪º╪¿┘ä ┘╛┘ê█î╪º ╪º┘å╪¬╪«╪º╪¿ ╪¿█î┘å ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ┘╛┘ê█î╪º ╪┤╪º┘à┘ä ┘à╪¿╪º╪»┘ä╪º╪¬ ╪¿┘å█î╪º╪»█î ╪º╪│╪¬ [53]: | ╪¼┘å╪¿┘ç | ╪º╪│╪¬╪º╪¬█î┌⌐ (C++23) | ┘╛┘ê█î╪º (Java/C#) | |ΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-| | ╪╣┘à┘ä┌⌐╪▒╪» | ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ | ┘ç╪▓█î┘å┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | ╪º┘å╪╣╪╖╪º┘ü┘╛╪░█î╪▒█î | ┘ü┘é╪╖ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä | ╪»╪│╪¬╪▒╪│█î ┌⌐╪º┘à┘ä ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | ╪º█î┘à┘å█î ┘å┘ê╪╣ | ┌⌐╪º┘à┘ä | ╪¿╪▒╪▒╪│█î┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î | ╪¬╪ú╪½█î╪▒ ╪¡╪»╪º┘é┘ä | ╪│╪▒╪¿╪º╪▒ ┘ü╪▒╪º╪»╪º╪»┘ç | | ╪º╪┤┌⌐╪º┘ä╪▓╪»╪º█î█î | ╪«╪╖╪º┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä | ╪┤┌⌐╪│╪¬┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | ### 2.4.2 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪▓╪¿╪º┘å ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪«╪¬┘ä┘ü █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘ä╪º╪¡╪╕╪º╪¬ ┘à┘ç┘à ╪╖╪▒╪º╪¡█î ╪▒╪º ╪ó╪┤┌⌐╪º╪▒ ┘à█î┌⌐┘å┘å╪» [54]: **█î┌⌐┘╛╪º╪▒┌å┌»█î ┘å╪¡┘ê█î:** ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪º ╪º┘ä┌»┘ê┘ç╪º█î ┘å╪¡┘ê ┘à┘ê╪¼┘ê╪» C++ █î┌⌐┘╛╪º╪▒┌å┘ç ┘à█î╪┤┘ê╪»╪î ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪▓╪¿╪º┘å┘ç╪º█î█î ┘à╪º┘å┘å╪» C# ┘å█î╪º╪▓ ╪¿┘ç ┘å╪¡┘ê ┘ê█î┌ÿ┌»█î ╪«╪º╪╡ ╪»╪º╪▒┘å╪». **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪│█î╪│╪¬┘à ┘å┘ê╪╣:** ╪│█î╪│╪¬┘à ┘å┘ê╪╣ ╪º╪│╪¬╪º╪¬█î┌⌐ C++ ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪»╪î ╪¿╪▒╪«┘ä╪º┘ü ╪▓╪¿╪º┘å┘ç╪º█î ┘å┘ê╪╣ ┘╛┘ê█î╪º. **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪╣┘à┘ä┌⌐╪▒╪»:** ╪º╪╡┘ä ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ┘à┘ä╪º╪¡╪╕┘ç ╪»┘é█î┘é ╪¬╪ú╪½█î╪▒╪º╪¬ ╪╣┘à┘ä┌⌐╪▒╪» ╪»╪▒ ┘ç╪▒ ╪¬╪╡┘à█î┘à ╪╖╪▒╪º╪¡█î ╪▒╪º ╪º╪¼╪¿╪º╪▒ ┘à█î┌⌐┘å╪». ### 2.4.3 ╪¬╪ú╪½█î╪▒ ╪º┌⌐┘ê╪│█î╪│╪¬┘à ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪½╪▒╪º╪¬ ╪╣┘à█î┘é█î ╪¿╪▒ ╪º┌⌐┘ê╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪▒╪¿┘ê╪╖┘ç ╪»╪º╪▒┘å╪» [55]: **╪╖╪▒╪º╪¡█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç:** ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪│╪¬┘ç┘ç╪º█î ╪¼╪»█î╪»█î ╪º╪▓ ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º ┘ê ┌å╪º╪▒┌å┘ê╪¿┘ç╪º ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» ┌⌐┘ç ┘é╪¿┘ä╪º┘ï ╪║█î╪▒┘à┘à┌⌐┘å █î╪º ╪║█î╪▒╪╣┘à┘ä█î ╪¿┘ê╪»┘å╪». **╪┤█î┘ê┘ç┘ç╪º█î ╪¬┘ê╪│╪╣┘ç:** ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿┘ê╪»┘å ╪¿╪º╪▓╪¬╪º╪¿ ┘å╪¡┘ê┘ç ╪▒┘ê█î╪º╪▒┘ê█î█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪¿╪º ┘à╪│╪º╪ª┘ä ╪▒╪º ╪¬╪║█î█î╪▒ ┘à█î╪»┘ç╪» ┘ê ╪º╪║┘ä╪¿ ┘à┘å╪¼╪▒ ╪¿┘ç ╪│╪¿┌⌐┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º╪╣┘ä╪º┘å█î╪¬╪▒ ┘à█î╪┤┘ê╪». **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪º╪¿╪▓╪º╪▒:** ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î IDE╪î ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ╪¿╪▒┘å╪º┘à┘ç┘ç╪º█î ┌⌐┘à┌⌐█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¬╪ú╪½█î╪▒ ┘à█î┌»╪░╪º╪▒┘å╪». ## 2.5 ╪┤┌⌐╪º┘ü┘ç╪º█î ╪¬╪¡┘é█î┘é ┘ê ┘ü╪▒╪╡╪¬┘ç╪º ╪¬╪¡┘ä█î┘ä ┘à╪º ┌å┘å╪»█î┘å ╪¡┘ê╪▓┘ç ╪▒╪º ╪ó╪┤┌⌐╪º╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ╪¬╪¡┘é█î┘é ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘à█î╪¬┘ê╪º┘å╪» ┘à╪┤╪º╪▒┌⌐╪¬┘ç╪º█î ┘à┘ç┘à█î ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤╪» [56]: **╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪╣┘à┘ä┌⌐╪▒╪»:** ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪▓╪º█î╪º█î ┘å╪╕╪▒█î ╪╣┘à┘ä┌⌐╪▒╪» ┘ê╪º╪╢╪¡ ╪º╪│╪¬╪î ╪º╪▒╪▓█î╪º╪¿█î ╪¼╪º┘à╪╣ ╪╣┘à┘ä┌⌐╪▒╪» ╪»╪▒ ┘à┘ê╪º╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç ┘à╪¬┘å┘ê╪╣ ┘ç┘å┘ê╪▓ ┘à╪¡╪»┘ê╪» ╪¿╪º┘é█î ┘à╪º┘å╪»┘ç ╪º╪│╪¬. **╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º:** ╪¬╪º╪▓┌»█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿┘ç ╪º█î┘å ┘à╪╣┘å█î ╪º╪│╪¬ ┌⌐┘ç ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘ç┘å┘ê╪▓ ╪»╪▒ ╪¡╪º┘ä ╪╕┘ç┘ê╪▒ ┘ç╪│╪¬┘å╪». **╪º┘ä┌»┘ê┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î:** ╪¬╪╣╪º┘à┘ä ╪¿█î┘å ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ (┌⌐╪º┘å╪│┘╛╪¬┘ç╪º╪î ┌⌐┘ê╪▒┘ê╪¬█î┘å┘ç╪º╪î ┘à╪º┌ÿ┘ê┘ä┘ç╪º) ┘å█î╪º╪▓ ╪¿┘ç ╪¬╪¡┘é█î┘é ╪│█î╪│╪¬┘à╪º╪¬█î┌⌐ ╪»╪º╪▒╪». **╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘╛╪░█î╪▒╪┤:** ╪»╪▒┌⌐ ╪º█î┘å┌⌐┘ç ╪¬█î┘à┘ç╪º ┌å┌»┘ê┘å┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿┘ç ╪╖┘ê╪▒ ┘à╪ñ╪½╪▒ ╪º╪▓ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ╪¿┘ç ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪º┘å╪¬┘é╪º┘ä █î╪º╪¿┘å╪»╪î ┘å┌»╪▒╪º┘å█î ╪╣┘à┘ä█î ┘à┘ç┘à█î ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪». # 3. ┌å╪º╪▒┌å┘ê╪¿ ╪¿╪º╪▓╪¬╪º╪¿ C++23 ## 3.1 ┘à┘ü╪º┘ç█î┘à ╪º╪╡┘ä█î ╪¿╪º╪▓╪¬╪º╪¿ ### 3.1.1 ┘╛╪▒┘ê╪¬┌⌐┘ä ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒ ╪º╪│╪º╪│ █î┌⌐ ┘╛╪▒┘ê╪¬┌⌐┘ä ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ ┘╛█î┌å█î╪»┘ç ╪│╪º╪«╪¬┘ç ╪┤╪»┘ç ┌⌐┘ç ╪»╪│╪¬╪▒╪│█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿┘ç ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» [57]. ╪»╪▒ ┘ç╪│╪¬┘ç ╪ó┘å╪î ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘ü┘ç┘ê┘à **┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º** ╪▒╪º ┘à╪╣╪▒┘ü█î ┘à█î┌⌐┘å╪» - ┘å┘à╪º█î╪┤┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ╪▓╪¿╪º┘å ┘à╪º┘å┘å╪» ╪º┘å┘ê╪º╪╣╪î ╪¬┘ê╪º╪¿╪╣╪î ┘à╪¬╪║█î╪▒┘ç╪º ┘ê namespace ┘ç╪º. ╪¿┘ä┘ê┌⌐ ╪│╪º╪«╪¬┘à╪º┘å█î ╪¿┘å█î╪º╪»█î std::meta::info ╪º╪│╪¬╪î █î┌⌐ ┘å┘ê╪╣ handle ┘à╪¿┘ç┘à ┌⌐┘ç ┘à┘ê╪¼┘ê╪»█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪┤╪»┘ç ╪▒╪º ┘å┘à╪º█î╪┤ ┘à█î╪»┘ç╪»: cpp #include <experimental/reflect> using namespace std::experimental::reflect; struct Person { std::string name; int age; void greet() const; }; // ╪¿┘ç ╪»╪│╪¬ ╪ó┘ê╪▒╪»┘å ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ ╪¿╪▒╪º█î ┘å┘ê╪╣ Person constexpr auto person\_meta = reflexpr(Person); static\_assert(std::is\_same\_v<decltype(person\_meta), const std::meta::info>); ╪¿╪▒╪«┘ä╪º┘ü ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º╪î ╪º█î┘å ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º ┘ü┘é╪╖ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê╪¼┘ê╪» ╪»╪º╪▒┘å╪» ┘ê ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º ╪¬╪╢┘à█î┘å ┘à█î┌⌐┘å┘å╪» [58]. ╪│█î╪│╪¬┘à ┘å┘ê╪╣ ╪º╪╖┘à█î┘å╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪╣┘à┘ä█î╪º╪¬ ┘å╪º┘à╪╣╪¬╪¿╪▒ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿┘ç ╪¼╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪┤┘å╪º╪│╪º█î█î ┘à█î╪┤┘ê┘å╪». ### 3.1.2 ┌⌐┘ê╪ª╪▒█î┘ç╪º ┘ê ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ API ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪¼┘à┘ê╪╣┘ç ╪║┘å█î ╪º╪▓ ╪¬┘ê╪º╪¿╪╣ ┌⌐┘ê╪ª╪▒█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» ┌⌐┘ç ╪▒┘ê█î ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º ╪╣┘à┘ä ┘à█î┌⌐┘å┘å╪» ╪¬╪º ╪º╪╖┘ä╪º╪╣╪º╪¬█î ╪»╪▒╪¿╪º╪▒┘ç ┘à┘ê╪¼┘ê╪»█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪┤╪»┘ç ╪º╪│╪¬╪«╪▒╪º╪¼ ┌⌐┘å┘å╪». ╪º█î┘å ┌⌐┘ê╪ª╪▒█î┘ç╪º ╪º╪▓ ╪º┘ä┌»┘ê█î ┘å╪º┘à┌»╪░╪º╪▒█î ╪│╪º╪▓┌»╪º╪▒ ┘╛█î╪▒┘ê█î ┘à█î┌⌐┘å┘å╪» ┘ê █î╪º ┘à┘é╪º╪»█î╪▒ constexpr █î╪º ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ╪º╪╢╪º┘ü█î ╪¿╪▒┘à█î┌»╪▒╪»╪º┘å┘å╪»: cpp // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê╪╣ ╪º┘ê┘ä█î┘ç constexpr bool is\_class\_type = is\_class\_v<person\_meta>; constexpr auto type\_name = get\_name\_v<person\_meta>; constexpr size\_t type\_size = get\_size\_v<person\_meta>; // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪»╪│╪¬╪▒╪│█î ╪╣╪╢┘ê constexpr auto data\_members = get\_data\_members\_t<person\_meta>{}; constexpr auto member\_functions = get\_member\_functions\_t<person\_meta>{}; constexpr auto constructors = get\_constructors\_t<person\_meta>{}; // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪▒╪º╪¿╪╖┘ç constexpr auto base\_classes = get\_base\_classes\_t<person\_meta>{}; constexpr bool is\_polymorphic = is\_polymorphic\_v<person\_meta>; ### 3.1.3 ╪│┌⌐╪º┘å╪│┘ç╪º█î ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ █î┌⌐█î ╪º╪▓ ┘é╪»╪▒╪¬┘à┘å╪»╪¬╪▒█î┘å ╪¼┘å╪¿┘ç┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒╪«┘ê╪▒╪» ╪ó┘å ╪¿╪º ┘à╪¼┘à┘ê╪╣┘ç┘ç╪º█î ┘à┘ê╪¼┘ê╪»█î╪¬┘ç╪º█î ┘à╪▒╪¬╪¿╪╖ ╪º╪│╪¬. ╪¿┘ç ╪¼╪º█î ╪¿╪▒┌»╪▒╪»╪º┘å╪»┘å ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ╪│┘å╪¬█î╪î ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ **╪│┌⌐╪º┘å╪│┘ç╪º█î ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬** ╪¿╪▒┘à█î┌»╪▒╪»╪º┘å┘å╪» - ╪│┌⌐╪º┘å╪│┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┌⌐┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┌»╪│╪¬╪▒╪┤ template parameter pack ┘╛╪▒╪»╪º╪▓╪┤ ╪┤┘ê┘å╪» [59]: cpp template<typename T> void print\_member\_info() { constexpr auto meta\_type = reflexpr(T); constexpr auto members = get\_data\_members\_t<meta\_type>{}; // ╪¬┌⌐╪▒╪º╪▒ ╪▒┘ê█î ╪º╪╣╪╢╪º ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ fold expressions (C++17) []<auto... Ms>(std::index\_sequence<Ms...>) { ((std::cout << "Member " << Ms << ": " << get\_name\_v<get\_element\_v<Ms, decltype(members)>> << " (type: " << get\_display\_name\_v<get\_type\_t<get\_element\_v<Ms, decltype(members)>>> << ")\n"), ...); }(std::make\_index\_sequence<get\_size\_v<members>>{}); } ╪º█î┘å ╪▒┘ê█î┌⌐╪▒╪» ╪¬┌⌐╪▒╪º╪▒ ┌⌐╪º╪▒╪ó┘à╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪»┘ê┘å ╪│╪▒╪¿╪º╪▒ ┘à╪▒╪¬╪¿╪╖ ╪¿╪º ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º █î╪º ╪¿╪º╪▓┌»╪┤╪¬ ┘╛█î┌å█î╪»┘ç ┘é╪º┘ä╪¿ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» [60]. ## 3.2 ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î API ╪¿╪º╪▓╪¬╪º╪¿ ### 3.2.1 ╪º█î┘à┘å█î ┘å┘ê╪╣ ┘ê ╪¬╪ú█î█î╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä API ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪▒╪▒╪│█î ┘é┘ê█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ê┘ä┘ê█î╪¬ ┘à█î╪»┘ç╪». ╪¬┘à╪º┘à ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪╖┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘à█î╪┤┘ê┘å╪» ┘ê ╪º╪▓ ╪«╪╖╪º┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º█î╪¼ ╪»╪▒ ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘å┘ê╪╣ ┘╛┘ê█î╪º ╪¼┘ä┘ê┌»█î╪▒█î ┘à█î┌⌐┘å┘å╪» [61]: cpp template<std::meta::info Member> constexpr auto get\_member\_value(const auto& obj) requires std::meta::is\_data\_member(Member) { // ╪¬╪ú█î█î╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪╖┘à█î┘å╪º┘å ┘à█î╪»┘ç╪» Member ┘ê╪º┘é╪╣╪º┘ï █î┌⌐ data member ╪º╪│╪¬ return obj.\*(std::meta::get\_pointer\_v<Member>); } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º ╪º█î┘à┘å█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä struct Point { int x, y; }; constexpr auto point\_meta = reflexpr(Point); constexpr auto x\_member = get\_element\_v<0, get\_data\_members\_t<point\_meta>>; Point p{10, 20}; auto x\_value = get\_member\_value<x\_member>(p); // ╪»╪│╪¬╪▒╪│█î ╪º█î┘à┘å ┘å┘ê╪╣ ### 3.2.2 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à┘ê╪¼┘ê╪» ╪▓╪¿╪º┘å ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î █î┌⌐┘╛╪º╪▒┌å┘ç ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à┘ê╪¼┘ê╪» C++╪î ╪¿┘ç ┘ê█î┌ÿ┘ç ┘é╪º┘ä╪¿┘ç╪º ┘ê ┌⌐╪º┘å╪│┘╛╪¬┘ç╪º ╪╖╪▒╪º╪¡█î ╪┤╪»┘ç ╪º╪│╪¬ [62]. ╪º█î┘å █î┌⌐┘╛╪º╪▒┌å┌»█î ╪º┘ä┌»┘ê┘ç╪º█î ╪¬╪▒┌⌐█î╪¿ ┘é╪»╪▒╪¬┘à┘å╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪»: cpp template<typename T> concept Reflectable = requires { reflexpr(T); typename get\_data\_members\_t<reflexpr(T)>; }; template<Reflectable T> std::string to\_json(const T& obj) { // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┌⌐╪º┘å╪│┘╛╪¬ return detail::serialize\_impl(obj, reflexpr(T)); } ### 3.2.3 ╪╖╪▒╪º╪¡█î ╪╣┘à┘ä┌⌐╪▒╪»-┘à╪¡┘ê╪▒ ┘ç╪▒ ╪¼┘å╪¿┘ç ╪º╪▓ API ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¿┘ç ╪¡╪»╪º┘é┘ä ╪▒╪│╪º┘å╪»┘å ╪│╪▒╪¿╪º╪▒ ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¬╪╢┘à█î┘å ┘ç╪▓█î┘å┘ç ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪╖╪▒╪º╪¡█î ╪┤╪»┘ç ╪º╪│╪¬ [63]. ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪º╪▒╪▓█î╪º╪¿█î constexpr ┘ê ┌»╪│╪¬╪▒╪┤ template parameter pack ┘å█î╪º╪▓ ╪¿┘ç dispatch ╪▓┘à╪º┘å ╪º╪¼╪▒╪º █î╪º ┘ü╪▒╪º╪«┘ê╪º┘å█î┘ç╪º█î ╪¬╪º╪¿╪╣ ┘à╪¼╪º╪▓█î ╪▒╪º ╪¡╪░┘ü ┘à█î┌⌐┘å╪»: cpp // ╪¿╪º╪▓╪¬╪º╪¿ ╪│┘å╪¬█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º (╪│╪¿┌⌐ ╪¼╪º┘ê╪º) // Object field = obj.getClass().getField("name"); // String value = (String) field.get(obj); // Dispatch ╪▓┘à╪º┘å ╪º╪¼╪▒╪º // ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä C++23 template<auto Member> constexpr auto get\_field\_value(const auto& obj) { return obj.\*(get\_pointer\_v<Member>); // ╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à ╪¡╪º┘ü╪╕┘ç } ## 3.3 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪│█î╪│╪¬┘à ┘é╪º┘ä╪¿ ┘à┘ê╪¼┘ê╪» ### 3.3.1 ╪¿┘ç╪¿┘ê╪» ╪º╪│╪¬┘å╪¬╪º╪¼ ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬┘å╪¬╪º╪¼ ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ ╪▒╪º ╪¿╪º ╪º╪▒╪º╪ª┘ç ╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ ╪¿┘ç╪¿┘ê╪» ┘à█î╪¿╪«╪┤╪» ┘ê ┘å█î╪º╪▓ ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┘╛█î┌å█î╪»┘ç SFINAE ╪▒╪º ╪¡╪░┘ü ┘à█î┌⌐┘å╪» [64]: cpp // ╪▒┘ê█î┌⌐╪▒╪» ╪│┘å╪¬█î SFINAE template<typename T> auto serialize\_impl(const T& obj) -> std::enable\_if\_t< std::conjunction\_v< std::is\_default\_constructible<T>, std::is\_copy\_constructible<T>, has\_member\_serialize<T> >, std::string> { // ┘à┘å╪╖┘é ╪º╪│╪¬┘å╪¬╪º╪¼ ┘╛█î┌å█î╪»┘ç } // ╪▒┘ê█î┌⌐╪▒╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> std::string serialize(const T& obj) requires requires { reflexpr(T); } { // ╪¬╪¡┘ä█î┘ä ┘à╪│╪¬┘é█î┘à ┘å┘ê╪╣ ╪¿╪»┘ê┘å ┘à╪º╪┤█î┘å╪ó┘ä╪º╪¬ ┘╛█î┌å█î╪»┘ç ┘é╪º┘ä╪¿ constexpr auto members = get\_data\_members\_t<reflexpr(T)>{}; return serialize\_members(obj, members); } ### 3.3.2 ╪¿┘ç╪¿┘ê╪» ┘é╪º┘ä╪¿ Variadic ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪«┘ê╪¿█î ╪¿╪º ┘é╪º┘ä╪¿┘ç╪º█î variadic ┌⌐╪º╪▒ ┘à█î┌⌐┘å╪» ┘ê ╪º┘ä┌»┘ê┘ç╪º█î ┘é╪»╪▒╪¬┘à┘å╪» ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» [65]: cpp template<typename... Types> class variant\_serializer { template<typename T> static std::string serialize\_variant(const std::variant<Types...>& var) { if (std::holds\_alternative<T>(var)) { return serialize\_reflected\_type(std::get<T>(var)); } return serialize\_next\_type</\* next type \*/>(var); } template<typename T> static std::string serialize\_reflected\_type(const T& obj) { constexpr auto meta = reflexpr(T); // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪»┘ê┘å ╪¬╪«╪╡╪╡█î╪º╪¿█î ╪╡╪▒█î╪¡ return reflect\_serialize(obj, meta); } }; ### 3.3.3 ┌⌐╪º┘ç╪┤ ╪¬╪«╪╡╪╡█î╪º╪¿█î ┘é╪º┘ä╪¿ █î┌⌐█î ╪º╪▓ ┘à┘ç┘à╪¬╪▒█î┘å ┘à╪▓╪º█î╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┌⌐╪º┘ç╪┤ ┌å╪┤┘à┌»█î╪▒ ╪¬╪«╪╡╪╡█î╪º╪¿█î┘ç╪º█î ┘é╪º┘ä╪¿ ┘à┘ê╪▒╪» ┘å█î╪º╪▓ ╪º╪│╪¬ [66]. █î┌⌐ type trait ╪▒╪º ╪»╪▒ ┘å╪╕╪▒ ╪¿┌»█î╪▒█î╪» ┌⌐┘ç ╪¬╪┤╪«█î╪╡ ┘à█î╪»┘ç╪» ╪ó█î╪º ┘å┘ê╪╣█î ╪╣╪╢┘ê ╪«╪º╪╡█î ╪»╪º╪▒╪»: cpp // ╪▒┘ê█î┌⌐╪▒╪» ╪│┘å╪¬█î: ┘å█î╪º╪▓ ╪¿┘ç ╪¬╪«╪╡╪╡█î╪º╪¿█î ╪╡╪▒█î╪¡ █î╪º SFINAE ┘╛█î┌å█î╪»┘ç template<typename T, typename = void> struct has\_to\_string : std::false\_type {}; template<typename T> struct has\_to\_string<T, std::void\_t<decltype(std::declval<T>().to\_string())>> : std::true\_type {}; // ╪▒┘ê█î┌⌐╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿: ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪╣┘à┘ê┘à█î ┘ê╪º╪¡╪» template<typename T> constexpr bool has\_to\_string\_v = []() { constexpr auto meta = reflexpr(T); constexpr auto functions = get\_member\_functions\_t<meta>{}; return []<auto... Fs>(std::index\_sequence<Fs...>) { return ((get\_name\_v<get\_element\_v<Fs, decltype(functions)>> == "to\_string") || ...); }(std::make\_index\_sequence<get\_size\_v<functions>>{}); }(); ## 3.4 ╪¬╪¡┘ä█î┘ä ┘å╪¡┘ê█î ┘ê ┘à╪╣┘å╪º█î█î ### 3.4.1 ╪╣┘à┘ä┌»╪▒ reflexpr ╪╣┘à┘ä┌»╪▒ reflexpr ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘å┘é╪╖┘ç ┘ê╪▒┘ê╪» ╪º╪╡┘ä█î ╪¿┘ç ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪╣┘à┘ä ┘à█î┌⌐┘å╪» [67]. ╪º┘å┘ê╪º╪╣ ┘à╪«╪¬┘ä┘ü ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ╪▓╪¿╪º┘å ╪▒╪º ┘à█î┘╛╪░█î╪▒╪» ┘ê ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ┘à╪▒╪¿┘ê╪╖┘ç ╪▒╪º ╪¿╪▒┘à█î┌»╪▒╪»╪º┘å╪»: cpp // ╪¿╪º╪▓╪¬╪º╪¿ ┘å┘ê╪╣ constexpr auto type\_meta = reflexpr(int); constexpr auto class\_meta = reflexpr(std::string); // ╪¿╪º╪▓╪¬╪º╪¿ namespace constexpr auto std\_meta = reflexpr(std); // ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪¬╪║█î╪▒ int global\_var = 42; constexpr auto var\_meta = reflexpr(global\_var); // ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪º╪¿╪╣ void my\_function(int, double); constexpr auto func\_meta = reflexpr(my\_function); ╪╣┘à┘ä┌»╪▒ ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘å╪¼╪º┘à ┘à█î╪»┘ç╪» ╪¬╪º ╪º╪╖┘à█î┘å╪º┘å ╪¡╪º╪╡┘ä ┌⌐┘å╪» ┌⌐┘ç ╪ó╪▒┌»┘ê┘à╪º┘å ╪º╪▒╪º╪ª┘ç ╪┤╪»┘ç █î┌⌐ ┘ç╪»┘ü ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪╣╪¬╪¿╪▒ ╪º╪│╪¬ [68]. ### 3.4.2 ┘╛╪▒┘ê╪¬┌⌐┘ä┘ç╪º█î ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ┌å┘å╪»█î┘å ╪»╪│╪¬┘ç ╪º╪▓ ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º ╪▒╪º ╪¬╪╣╪▒█î┘ü ┘à█î┌⌐┘å╪»╪î ┘ç╪▒ ┌⌐╪»╪º┘à ╪¿╪º ╪▒╪º╪¿╪╖┘ç╪º█î ┌⌐┘ê╪ª╪▒█î ╪«╪º╪╡ [69]: **┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ┘å┘ê╪╣:** cpp template<std::meta::info TypeMeta> requires std::meta::is\_type(TypeMeta) class type\_analyzer { static constexpr bool is\_fundamental = std::meta::is\_fundamental\_v<TypeMeta>; static constexpr bool is\_class = std::meta::is\_class\_v<TypeMeta>; static constexpr auto name = std::meta::get\_name\_v<TypeMeta>; static constexpr auto size = std::meta::get\_size\_v<TypeMeta>; }; **┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ╪╣╪╢┘ê:** cpp template<std::meta::info MemberMeta> requires std::meta::is\_data\_member(MemberMeta) class member\_analyzer { static constexpr auto name = std::meta::get\_name\_v<MemberMeta>; static constexpr auto type = std::meta::get\_type\_t<MemberMeta>; static constexpr auto offset = std::meta::get\_offset\_v<MemberMeta>; static constexpr bool is\_public = std::meta::is\_public\_v<MemberMeta>; }; ### 3.4.3 ╪▓┘à█î┘å┘ç ╪º╪▒╪▓█î╪º╪¿█î Constexpr ╪¬┘à╪º┘à ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪▓┘à█î┘å┘ç┘ç╪º█î ╪º╪▒╪▓█î╪º╪¿█î constexpr ╪▒╪« ┘à█î╪»┘ç┘å╪» ┘ê ╪º╪¼╪▒╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¬╪╢┘à█î┘å ┘à█î┌⌐┘å┘å╪» [70]. ╪º█î┘å ╪º┘ä╪▓╪º┘à ┌å┘å╪»█î┘å ╪¬╪╡┘à█î┘à ╪╖╪▒╪º╪¡█î ╪▒╪º ┘ç╪»╪º█î╪¬ ┘à█î┌⌐┘å╪»: cpp template<typename T> constexpr auto analyze\_type() { constexpr auto meta = reflexpr(T); // ╪¬┘à╪º┘à ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º█î╪» constexpr ╪¿╪º╪┤┘å╪» constexpr auto member\_count = get\_size\_v<get\_data\_members\_t<meta>>; constexpr auto is\_trivial = is\_trivially\_copyable\_v<meta>; struct analysis\_result { size\_t members; bool trivial; std::string\_view name; }; return analysis\_result{ .members = member\_count, .trivial = is\_trivial, .name = get\_name\_v<meta> }; } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä constexpr auto person\_analysis = analyze\_type<Person>(); static\_assert(person\_analysis.members > 0); ## 3.5 ╪º┘ä┌»┘ê┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪º╪▓╪¬╪º╪¿ ### 3.5.1 ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤╪▒╪╖█î ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤╪▒╪╖█î ┘╛█î┌å█î╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘ê╪º┘é╪╣█î ┘å┘ê╪╣ ╪¿┘ç ╪¼╪º█î ╪¬╪«╪╡╪╡█î╪º╪¿█î┘ç╪º█î ╪┤┌⌐┘å┘å╪»┘ç ┘é╪º┘ä╪¿ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» [71]: cpp template<typename T> auto serialize(const T& obj) { constexpr auto meta = reflexpr(T); if constexpr (has\_custom\_serializer\_v<T>) { return obj.serialize(); } else if constexpr (is\_container\_v<meta>) { return serialize\_container(obj, meta); } else if constexpr (is\_arithmetic\_v<meta>) { return serialize\_arithmetic(obj); } else { return serialize\_aggregate(obj, meta); } } ### 3.5.2 ╪¬╪╖╪¿█î┘é ┘å┘ê╪╣ ┘ê ╪¬┘ê┘ä█î╪» ┘╛╪▒╪º┌⌐╪│█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐┘ä╪º╪│┘ç╪º█î ╪ó╪»╪º┘╛╪¬┘ê╪▒ ┘ê ┘╛╪▒╪º┌⌐╪│█î ╪▒╪º ╪¬╪│┘ç█î┘ä ┘à█î┌⌐┘å╪» [72]: cpp template<typename Interface> class reflection\_proxy { std::any target\_; public: template<typename Implementation> reflection\_proxy(Implementation&& impl) : target\_(std::forward<Implementation>(impl)) {} // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê╪º╪¿╪╣ forwarding ╪¿╪▒╪º█î ╪¬┘à╪º┘à ┘à╪¬╪»┘ç╪º█î ╪▒╪º╪¿╪╖ template<auto Method> requires std::meta::is\_member\_function(Method) auto invoke(auto&&... args) { constexpr auto method\_name = get\_name\_v<Method>; constexpr auto return\_type = get\_return\_type\_t<Method>; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î █î╪º┘ü╪¬┘å ┘ê ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪¬╪» ┘à╪▒╪¿┘ê╪╖┘ç ╪▒┘ê█î target return invoke\_by\_name<method\_name>(std::any\_cast<auto&>(target\_), std::forward<decltype(args)>(args)...); } }; ### 3.5.3 ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º┘ä┌»┘ê█î ╪╖╪▒╪º╪¡█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º╪▓╪¬╪º╪¿ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ╪▒╪º█î╪¼ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» [73]: cpp template<typename T> class auto\_visitor { // ╪¬┘ê┘ä█î╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º┘ä┌»┘ê█î visitor ╪¿╪▒ ╪º╪│╪º╪│ ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘å┘ê╪╣ static\_assert(std::meta::is\_polymorphic\_v<reflexpr(T)>); template<typename Visitor> static auto visit(const T& obj, Visitor&& visitor) { constexpr auto derived\_types = get\_derived\_types\_t<reflexpr(T)>{}; return visit\_impl(obj, std::forward<Visitor>(visitor), derived\_types); } private: template<typename Visitor, auto... DerivedMetas> static auto visit\_impl(const T& obj, Visitor&& visitor, std::index\_sequence<DerivedMetas...>) { // ╪¬┘ê┘ä█î╪» dispatch ┘╛┘ê█î╪º█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ typeid const std::type\_info& runtime\_type = typeid(obj); auto result = std::optional<decltype(visitor(std::declval<T>()))>{}; ((runtime\_type == typeid(get\_reflected\_type\_t<DerivedMetas>) ? (result = visitor(static\_cast<const get\_reflected\_type\_t<DerivedMetas>&>(obj)), true) : false) || ...); return \*result; } }; ## 3.6 ┘à┘ä╪º╪¡╪╕╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ### 3.6.1 █î┌⌐┘╛╪º╪▒┌å┌»█î ┘ü╪º╪▓ ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘å█î╪º╪▓ ╪¿┘ç █î┌⌐┘╛╪º╪▒┌å┌»█î ╪»┘é█î┘é ╪¿╪º ╪«╪╖ ┘ä┘ê┘ä┘ç ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪º╪▒╪» [74]. ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º ╪¿╪º█î╪» ╪»╪▒ ╪╖┘ê┘ä ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ┘é╪º┘ä╪¿ ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿╪º╪┤┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪º╪╡┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪¼╪»╪º┌»╪º┘å┘ç ╪▒╪º ╪¡┘ü╪╕ ┌⌐┘å┘å╪»: cpp // ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¿╪º█î╪» ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬ ╪▒╪º ╪▒╪»█î╪º╪¿█î ┌⌐┘å╪» template<typename T> constexpr auto get\_serialization\_info() { constexpr auto meta = reflexpr(T); // ┘ê╪º╪¿╪│╪¬┌»█î ╪¿┘ç ╪¬╪╣╪▒█î┘ü ┌⌐╪º┘à┘ä T return analyze\_serialization\_requirements(meta); // ╪¿╪º█î╪» ╪¿╪▒╪º█î ┘å┘à┘ê┘å┘ç╪│╪º╪▓█î ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿╪º╪┤╪» } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪»╪▒ ┘ê╪º╪¡╪» ╪¬╪▒╪¼┘à┘ç ╪¼╪»╪º┌»╪º┘å┘ç extern template auto get\_serialization\_info<MyClass>(); // ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪º╪╣┘ä╪º┘å forward ### 3.6.2 ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪»█î╪¿╪º┌» ┘ê █î┌⌐┘╛╪º╪▒┌å┌»█î ╪º╪¿╪▓╪º╪▒ ┘à╪¬╪º-╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º█î╪» ╪¿╪º ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» ┘ê ╪¬┘ê╪│╪╣┘ç █î┌⌐┘╛╪º╪▒┌å┘ç ╪┤┘ê┘å╪» [75]: cpp template<typename T> void debug\_print\_type\_info() { constexpr auto meta = reflexpr(T); // ╪¿█î┘ä╪»┘ç╪º█î ╪»█î╪¿╪º┌» ╪¿╪º█î╪» ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¡┘ü╪╕ ┌⌐┘å┘å╪» // ╪¿╪▒╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î IDE ┘ê ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» if constexpr (std::meta::is\_debug\_build()) { emit\_debug\_info(meta); } } ### 3.6.3 ┘ü╪▒╪╡╪¬┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å┘å╪» [76]: cpp template<typename T> std::string fast\_serialize(const T& obj) { constexpr auto meta = reflexpr(T); // ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘à█î╪¬┘ê╪º┘å╪» ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¡┘ä█î┘ä ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐┘å╪» if constexpr (is\_pod\_serializable\_v<meta>) { // ╪¬┘ê┘ä█î╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ memcpy return serialize\_pod(obj); } else { // ╪¬┘ê┘ä█î╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ü█î┘ä╪» ╪¿┘ç ┘ü█î┘ä╪» return serialize\_structured(obj, meta); } } ╪º█î┘å ╪¬╪¡┘ä█î┘ä ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘╛╪º█î┘ç╪º█î ╪¼╪º┘à╪╣╪î ╪º█î┘à┘å ┘å┘ê╪╣ ┘ê ┌⌐╪º╪▒╪ó┘à╪» ╪¿╪▒╪º█î ╪»╪▒┘ê┘å┌»╪▒█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪» ┌⌐┘ç ┌å┌»┘ê┘å┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒ ╪º█î┘å ┘╛╪º█î┘ç ╪¿┘å╪º ┘à█î╪┤┘ê┘å╪» ╪¬╪º ╪º┘ä┌»┘ê┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î ╪¡╪¬█î ┘é╪»╪▒╪¬┘à┘å╪»╪¬╪▒█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ╪│╪º╪▓┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 57-76 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à╪┤╪«╪╡╪º╪¬ ┘ü┘å█î ╪»┘é█î┘é╪î ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ê ┘à┘é╪º┘ä╪º╪¬ ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 4. ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º: ╪╖╪▒╪º╪¡█î ┘ê ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ## 4.1 ┘à┘ü┘ç┘ê┘à ┘ê ╪º┘å┌»█î╪▓┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ### 4.1.1 ┌å╪┤┘à╪º┘å╪»╪º╪▓ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç ╪º┘ê╪¼ ╪»┘ç┘ç┘ç╪º ╪¬╪¡┘é█î┘é ╪»╪▒ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î ┘ê ╪│┘å╪¬╪▓ ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ç╪│╪¬┘å╪» [77]. ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê╪º┘å╪º█î█î ╪»╪▒┘ê┘å┌»╪▒█î ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┌⌐╪» ┘à┘ê╪¼┘ê╪» ╪▒╪º ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪»╪î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º **╪¬┘ê┘ä█î╪»** ┌⌐╪» ╪¼╪»█î╪» ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º╪î ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ┘ê ╪º┘ä╪▓╪º┘à╪º╪¬ ╪«╪º╪╡ ╪»╪º┘à┘å┘ç ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓┘å╪». ╪¿█î┘å╪┤ ╪¿┘å█î╪º╪»█î ┘╛╪┤╪¬ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º█î┘å ╪º╪│╪¬ ┌⌐┘ç ╪¿╪│█î╪º╪▒█î ╪º╪▓ ╪º┘ä┌»┘ê┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪┤╪º┘à┘ä ┌⌐╪» ╪¬┌⌐╪▒╪º╪▒█î ┘ê ╪║█î╪▒╪╢╪▒┘ê╪▒█î ┘ç╪│╪¬┘å╪» ┌⌐┘ç ╪º╪▓ ╪º┘ä┌»┘ê┘ç╪º█î ┘é╪º╪¿┘ä ┘╛█î╪┤╪¿█î┘å█î ┘╛█î╪▒┘ê█î ┘à█î┌⌐┘å┘å╪» [78]. ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪▒╪º█î╪¼ ╪▒╪º ╪»╪▒ ┘å╪╕╪▒ ╪¿┌»█î╪▒█î╪»: - **┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«╪╡┘ê╪╡█î╪¬**: ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ getter ┘ç╪º╪î setter ┘ç╪º ┘ê ┘à┘å╪╖┘é ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î - **╪│╪▒█î╪º┘ä╪│╪º╪▓█î**: ╪¬╪¿╪»█î┘ä ╪«┘ê╪»┌⌐╪º╪▒ ╪¿┘ç/╪º╪▓ ┘ü╪▒┘à╪¬┘ç╪º█î JSON╪î XML █î╪º ╪¿╪º█î┘å╪▒█î - **╪º┘ä┌»┘ê█î Observer**: ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ╪¬╪║█î█î╪▒╪º╪¬ ┘ê╪╢╪╣█î╪¬ - **┘å┌»╪º╪┤╪¬ ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç**: ┘å┌»╪º╪┤╪¬ ╪│╪¿┌⌐ ORM ╪¿█î┘å ╪ó╪¿╪¼┌⌐╪¬┘ç╪º ┘ê ╪╖╪▒╪¡┘ê╪º╪▒┘ç┘ç╪º█î ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç - **┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪▒╪º╪¿╪╖**: ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ delegation ┘ê ┘╛╪▒╪º┌⌐╪│█î ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ╪¿┘ç ╪º█î┘å ┘à╪│╪º╪ª┘ä ╪┤╪º┘à┘ä █î╪º ┌⌐╪»┘å┘ê█î╪│█î ╪»╪│╪¬█î ┌»╪│╪¬╪▒╪»┘ç █î╪º ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪º╪│╪¬. ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┌»╪▓█î┘å┘ç ╪│┘ê┘à█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪»: **┘à╪┤╪«╪╡┘ç ╪º╪╣┘ä╪º┘å█î** ╪▒┘ü╪¬╪º╪▒ ┘à╪╖┘ä┘ê╪¿ ╪¿╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î [79]. ### 4.1.2 ┘ü┘ä╪│┘ü┘ç ┘ê ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ C++23 ╪º╪▓ ┌å┘å╪»█î┘å ╪º╪╡┘ä ┌⌐┘ä█î╪»█î ┘╛█î╪▒┘ê█î ┘à█î┌⌐┘å╪» [80]: **┘é╪╡╪» ╪º╪╣┘ä╪º┘å█î**: ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪º╪¼╪º╪▓┘ç ┘à█î╪»┘ç┘å╪» ╪¿█î╪º┘å ┌⌐┘å┘å╪» *┌å┘ç* ┘à█î╪«┘ê╪º┘ç┘å╪» ╪¿┘ç ╪¼╪º█î *┌å┌»┘ê┘å┘ç* ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐┘å┘å╪»: cpp // ┘à╪┤╪«╪╡┘ç ╪º╪╣┘ä╪º┘å█î class $serializable $observable Person { std::string name; int age; double salary; }; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - ┘à╪¬╪»┘ç╪º█î to\_json() / from\_json() // - ╪½╪¿╪¬/╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î Observer // - ╪»╪│╪¬╪▒╪│█î┌⌐┘å┘å╪»┘ç┘ç╪º█î ╪«╪╡┘ê╪╡█î╪¬ ╪¿╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î // - ╪╣┘à┘ä┌»╪▒┘ç╪º█î ╪¿╪▒╪º╪¿╪▒█î ┘ê ┘à┘é╪º█î╪│┘ç **┘é╪º╪¿┘ä█î╪¬ ╪¬╪▒┌⌐█î╪¿**: ┌å┘å╪»█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿┘ç █î┌⌐ ┘å┘ê╪╣ ╪º╪╣┘à╪º┘ä ╪┤┘ê┘å╪»╪î ╪¿╪º ┘à╪╣┘å╪º╪┤┘å╪º╪│█î ╪¬╪▒┌⌐█î╪¿ ╪¿┘ç ╪«┘ê╪¿█î ╪¬╪╣╪▒█î┘ü ╪┤╪»┘ç: cpp class $entity("users") $auditable $cacheable User { // ╪¬╪▒┌⌐█î╪¿ ┘å┌»╪º╪┤╪¬ ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç╪î ┘ä╪º┌» ╪¡╪│╪º╪¿╪▒╪│█î ┘ê ┌⌐╪┤ }; **╪º█î┘à┘å█î ┘å┘ê╪╣**: ╪¬┘à╪º┘à ╪¬╪¿╪»█î┘ä┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘å┘ê╪╣-╪¿╪▒╪▒╪│█î ┘ê ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘à█î╪┤┘ê┘å╪» ┘ê ╪º╪▓ ╪«╪╖╪º┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º█î╪¼ ╪»╪▒ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¼┘ä┘ê┌»█î╪▒█î ┘à█î┌⌐┘å┘å╪». **╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º**: ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪º╪▓ ┘å╪╕╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ┘é╪º╪¿┘ä ╪¬╪┤╪«█î╪╡ ╪º╪▓ ┌⌐╪» ╪»╪│╪¬┘å┘ê█î╪│ ┘å█î╪│╪¬. ### 4.1.3 ╪▒╪º╪¿╪╖┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪╖┘ê╪▒ ╪¿┘å█î╪º╪»█î ╪¿╪▒ ╪▓█î╪▒╪│╪º╪«╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê╪╢█î╪¡ ╪»╪º╪»┘ç ╪┤╪»┘ç ╪»╪▒ ╪¿╪«╪┤ 3 ╪¿┘å╪º ┘à█î╪┤┘ê┘å╪». ╪▒╪º╪¿╪╖┘ç ┘ç┘à╪▓█î╪│╪¬ ╪º╪│╪¬ [81]: cpp constexpr void serializable(std::meta::info target) { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ┘å┘ê╪╣ ┘ç╪»┘ü ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å╪» std::meta::compiler.require(std::meta::is\_class(target), "serializable can only be applied to classes"); // ╪¬┌⌐╪▒╪º╪▒ ╪▒┘ê█î ╪º╪╣╪╢╪º█î ╪»╪º╪»┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ for (auto member : std::meta::data\_members\_of(target)) { generate\_serialization\_code(member); } // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ std::meta::compiler.declare(target, generate\_to\_json\_method(target)); std::meta::compiler.declare(target, generate\_from\_json\_method(target)); } ## 4.2 ┘å╪¡┘ê ╪¬╪╣╪▒█î┘ü ┘à╪¬╪º┌⌐┘ä╪º╪│ ### 4.2.1 ╪º╪╣┘ä╪º┘å ┘╛╪º█î┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪¬┘ê╪º╪¿╪╣ constexpr ╪¬╪╣╪▒█î┘ü ┘à█î╪┤┘ê┘å╪» ┌⌐┘ç ╪▒┘ê█î ╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î std::meta::info ┘å┘à╪º█î╪┤╪»┘ç┘å╪»┘ç ┘å┘ê╪╣ ┘ç╪»┘ü ╪╣┘à┘ä ┘à█î┌⌐┘å┘å╪» [82]: cpp #include <experimental/meta> constexpr void property(std::meta::info target) { // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌⌐┘ç target █î┌⌐ ┌⌐┘ä╪º╪│ ╪º╪│╪¬ std::meta::compiler.require(std::meta::is\_class(target), "property metaclass requires a class"); // ╪¬┘ê┘ä█î╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«╪╡┘ê╪╡█î╪¬ for (auto member : std::meta::data\_members\_of(target)) { if (std::meta::is\_private(member)) { generate\_property\_accessors(target, member); } } } // ╪º╪│╪¬┘ü╪º╪»┘ç class $property Person { private: std::string name\_; // ╪¬┘ê┘ä█î╪» getName()╪î setName() int age\_; // ╪¬┘ê┘ä█î╪» getAge()╪î setAge() }; ### 4.2.2 ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘╛╪º╪▒╪º┘à╪¬╪▒█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î█î ╪¿╪▒╪º█î ╪┤╪«╪╡█î╪│╪º╪▓█î ╪▒┘ü╪¬╪º╪▒ ╪«┘ê╪» ╪¿┘╛╪░█î╪▒┘å╪» [83]: cpp constexpr void entity(std::meta::info target, std::string\_view table\_name = "", bool generate\_crud = true) { auto actual\_table = table\_name.empty() ? std::meta::get\_name\_v<target> : table\_name; // ╪¬┘ê┘ä█î╪» ┘å┌»╪º╪┤╪¬ ╪¼╪»┘ê┘ä generate\_table\_mapping(target, actual\_table); if (generate\_crud) { generate\_crud\_operations(target); } } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º class $entity("user\_accounts", true) User { int id; std::string username; std::string email; }; ### 4.2.3 ╪º╪╣┘à╪º┘ä ╪┤╪▒╪╖█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ┘à┘å╪╖┘é ╪┤╪▒╪╖█î ╪¿╪▒ ╪º╪│╪º╪│ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘å┘ê╪╣ ╪┤╪º┘à┘ä ╪┤┘ê┘å╪» [84]: cpp constexpr void smart\_serializable(std::meta::info target) { // ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘à╪«╪¬┘ä┘ü ╪¿╪▒ ╪º╪│╪º╪│ ┘╛█î┌å█î╪»┌»█î ┘å┘ê╪╣ auto members = std::meta::data\_members\_of(target); if (std::meta::get\_size\_v<members> <= 5 && all\_pod\_members(members)) { generate\_binary\_serialization(target); } else if (has\_string\_members(members)) { generate\_json\_serialization(target); } else { generate\_xml\_serialization(target); } // ┘ç┘à█î╪┤┘ç ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¬┘ê┘ä█î╪» ┌⌐┘å generate\_validation\_methods(target); } ## 4.3 ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ### 4.3.1 ╪▒╪º╪¿╪╖ ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪▒╪º╪¿╪╖ std::meta::compiler ┘à┌⌐╪º┘å█î╪▓┘à ╪º╪╡┘ä█î ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» [85]. ╪º█î┘å ╪▒╪º╪¿╪╖ ╪¿┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º╪¼╪º╪▓┘ç ╪¬╪▓╪▒█î┘é ╪º╪╣┘ä╪º┘å┘ç╪º█î ╪¼╪»█î╪» ╪¿┘ç ┘å┘ê╪╣ ┘ç╪»┘ü ╪▒╪º ┘à█î╪»┘ç╪»: cpp namespace std::meta { struct compiler\_interface { // ╪¬╪▓╪▒█î┘é █î┌⌐ ╪¬╪º╪¿╪╣ ╪╣╪╢┘ê ╪¼╪»█î╪» static constexpr void declare(info target, std::string\_view code); // ╪¬╪▓╪▒█î┘é █î┌⌐ ╪╣╪╢┘ê ╪»╪º╪»┘ç ╪¼╪»█î╪» static constexpr void declare\_member(info target, info type, std::string\_view name); // ╪º┘ä╪▓╪º┘à █î┌⌐ ╪┤╪▒╪╖ (assertion ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä) static constexpr void require(bool condition, std::string\_view message); // ╪¬┘ê┘ä█î╪» ┘╛█î╪º┘à┘ç╪º█î ╪¬╪┤╪«█î╪╡█î static constexpr void warn(std::string\_view message); static constexpr void error(std::string\_view message); }; } ### 4.3.2 ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º╪║┘ä╪¿ ╪º╪▓ ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘é╪º┘ä╪¿ ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪º█î┘à┘å ┘å┘ê╪╣ ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å┘å╪» [86]: cpp constexpr void comparable(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪╣┘à┘ä┌»╪▒┘ç╪º█î ┘à┘é╪º█î╪│┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ╪╣╪╢┘ê std::string equality\_impl = R"( bool operator==(const )" + std::meta::get\_name\_v<target> + R"(& other) const { return true)"; for (auto member : std::meta::data\_members\_of(target)) { auto member\_name = std::meta::get\_name\_v<member>; equality\_impl += " && (" + member\_name + " == other." + member\_name + ")"; } equality\_impl += R"(; } bool operator!=(const )" + std::meta::get\_name\_v<target> + R"(& other) const { return !(\*this == other); } auto operator<=>(const )" + std::meta::get\_name\_v<target> + R"(& other) const { // ┘à┘é╪º█î╪│┘ç ╪│┘ç╪╖╪▒┘ü┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ std::tie return std::tie()"; bool first = true; for (auto member : std::meta::data\_members\_of(target)) { if (!first) equality\_impl += ", "; equality\_impl += std::meta::get\_name\_v<member>; first = false; } equality\_impl += R"() <=> std::tie()"; first = true; for (auto member : std::meta::data\_members\_of(target)) { if (!first) equality\_impl += ", "; equality\_impl += "other." + std::meta::get\_name\_v<member>; first = false; } equality\_impl += ");"; equality\_impl += "\n}"; std::meta::compiler.declare(target, equality\_impl); } ### 4.3.3 ╪º┘ä┌»┘ê┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪│┘å╪¬╪▓ ┌⌐╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘╛█î┌å█î╪»┘ç ┘à┘à┌⌐┘å ╪º╪│╪¬ ┘å█î╪º╪▓ ╪¿┘ç ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤┘å╪» [87]: cpp constexpr void state\_machine(std::meta::info target, std::span<const state\_transition> transitions) { // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¬╪╣╪▒█î┘ü ┘à╪º╪┤█î┘å ╪¡╪º┘ä╪¬ validate\_state\_machine(target, transitions); // ╪¬┘ê┘ä█î╪» ╪┤┘à╪º╪▒╪┤ ╪¡╪º┘ä╪¬ generate\_state\_enum(target, transitions); // ╪¬┘ê┘ä█î╪» ╪¼╪»┘ê┘ä ╪º┘å╪¬┘é╪º┘ä generate\_transition\_table(target, transitions); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î ┘à╪º╪┤█î┘å ╪¡╪º┘ä╪¬ std::string machine\_impl = R"( private: State current\_state\_ = State::)" + get\_initial\_state(transitions) + R"(; public: State get\_state() const { return current\_state\_; } template<typename Event> bool process\_event(const Event& event) { auto new\_state = transition\_table\_.find({current\_state\_, typeid(Event)}); if (new\_state != transition\_table\_.end()) { auto old\_state = current\_state\_; current\_state\_ = new\_state->second; on\_state\_change(old\_state, current\_state\_, event); return true; } return false; } protected: virtual void on\_state\_change(State from, State to, const auto& event) {} )"; std::meta::compiler.declare(target, machine\_impl); } ## 4.4 ╪º┘ä┌»┘ê┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ### 4.4.1 ╪¬┘ê┘ä█î╪» ┘ê ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪▒╪º╪¿╪╖ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪▒╪º╪¿╪╖ ╪▒╪º ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┌⌐┘å┘å╪» [88]: cpp constexpr void rest\_api(std::meta::info target, std::string\_view base\_path) { // ╪¬┘ê┘ä█î╪» endpoint ┘ç╪º█î REST API ╪¿╪▒ ╪º╪│╪º╪│ ┘à╪¬╪»┘ç╪º█î ╪╣┘à┘ê┘à█î for (auto method : std::meta::member\_functions\_of(target)) { if (std::meta::is\_public(method)) { auto method\_name = std::meta::get\_name\_v<method>; auto return\_type = std::meta::get\_return\_type\_t<method>; auto parameters = std::meta::get\_parameters\_t<method>; if (method\_name.starts\_with("get")) { generate\_get\_endpoint(target, method, base\_path); } else if (method\_name.starts\_with("create") || method\_name.starts\_with("add")) { generate\_post\_endpoint(target, method, base\_path); } else if (method\_name.starts\_with("update")) { generate\_put\_endpoint(target, method, base\_path); } else if (method\_name.starts\_with("delete") || method\_name.starts\_with("remove")) { generate\_delete\_endpoint(target, method, base\_path); } } } // ╪¬┘ê┘ä█î╪» ╪¼╪»┘ê┘ä ┘à╪│█î╪▒█î╪º╪¿█î generate\_routing\_table(target, base\_path); } class $rest\_api("/api/users") UserService { public: User getUser(int id); // ╪¬┘ê┘ä█î╪» GET /api/users/{id} User createUser(const User&); // ╪¬┘ê┘ä█î╪» POST /api/users void updateUser(int id, const User&); // ╪¬┘ê┘ä█î╪» PUT /api/users/{id} void deleteUser(int id); // ╪¬┘ê┘ä█î╪» DELETE /api/users/{id} }; ### 4.4.2 ╪¬┘ê┘ä█î╪» ┌å╪º╪▒┌å┘ê╪¿ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪▒╪º ┘à╪»█î╪▒█î╪¬ ┌⌐┘å┘å╪» [89]: cpp constexpr void serializable(std::meta::info target, serialization\_format format = json, naming\_convention naming = snake\_case) { // ╪¬┘ê┘ä█î╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«╪º╪╡ ┘ü╪▒┘à╪¬ switch (format) { case json: generate\_json\_serialization(target, naming); break; case xml: generate\_xml\_serialization(target, naming); break; case binary: generate\_binary\_serialization(target); break; case protobuf: generate\_protobuf\_serialization(target, naming); break; } // ╪¬┘ê┘ä█î╪» ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪╖╪▒╪¡┘ê╪º╪▒┘ç generate\_schema\_validation(target, format); // ╪¬┘ê┘ä█î╪» ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘å╪│╪«┘ç╪¿┘å╪»█î generate\_version\_handling(target, format); } // ┘à╪½╪º┘ä ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘╛█î┌å█î╪»┘ç class $serializable(json, snake\_case) $versioned(2) Person { std::string full\_name; // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿┘ç ╪╣┘å┘ê╪º┘å "full\_name" std::optional<int> age; // ┘à╪»█î╪▒█î╪¬ ┘ü█î┘ä╪» ╪º╪«╪¬█î╪º╪▒█î std::vector<std::string> tags; // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪ó╪▒╪º█î┘ç // ╪╖╪▒╪¡┘ê╪º╪▒┘ç ╪«┘ê╪»┌⌐╪º╪▒: {"full\_name": "string", "age": "int?", "tags": ["string"]} // ┘à╪»█î╪▒█î╪¬ ┘å╪│╪«┘ç: ┘à┘ç╪º╪¼╪▒╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪▓ v1 ╪¿┘ç v2 }; ### 4.4.3 ORM ┘ê █î┌⌐┘╛╪º╪▒┌å┌»█î ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪¼┘ç╪¬┌»█î╪▒█î ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘╛█î┌å█î╪»┘ç ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» [90]: cpp constexpr void entity(std::meta::info target, std::string\_view table\_name, database\_dialect dialect = postgresql) { // ╪¬┘ê┘ä█î╪» ╪╖╪▒╪¡┘ê╪º╪▒┘ç ╪¼╪»┘ê┘ä generate\_create\_table\_sql(target, table\_name, dialect); // ╪¬┘ê┘ä█î╪» ╪╣┘à┘ä█î╪º╪¬ CRUD generate\_find\_methods(target, table\_name, dialect); generate\_save\_method(target, table\_name, dialect); generate\_delete\_method(target, table\_name, dialect); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î query builder for (auto member : std::meta::data\_members\_of(target)) { if (std::meta::has\_attribute<indexed>(member)) { generate\_find\_by\_method(target, member, table\_name, dialect); } } // ╪¬┘ê┘ä█î╪» ┘à╪»█î╪▒█î╪¬ ╪▒╪º╪¿╪╖┘ç generate\_relationship\_methods(target, table\_name, dialect); } class $entity("users", postgresql) User { $primary\_key int id; $indexed $unique std::string email; std::string name; $nullable std::optional<std::string> bio; $one\_to\_many("user\_id") std::vector<Post> posts; $many\_to\_one Profile profile; // ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // static User find(int id); // static std::vector<User> find\_by\_email(const std::string& email); // static std::vector<User> find\_all(); // void save(); // void delete(); // std::vector<Post> get\_posts(); // Profile get\_profile(); }; ### 4.4.4 ╪º╪¬┘ê┘à╪º╪│█î┘ê┘å ╪º┘ä┌»┘ê█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘╛█î┌å█î╪»┘ç ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐┘å┘å╪» [91]: cpp constexpr void observer(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪▓█î╪▒╪│╪º╪«╪¬ observer std::string observer\_code = R"( private: mutable std::vector<std::function<void(const std::string&)>> observers\_; public: void add\_observer(std::function<void(const std::string&)> observer) { observers\_.push\_back(std::move(observer)); } void remove\_observer(const std::function<void(const std::string&)>& observer) { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪¿╪▒╪º█î ╪¡╪░┘ü observer } protected: void notify\_observers(const std::string& property\_name) const { for (const auto& observer : observers\_) { observer(property\_name); } } )"; std::meta::compiler.declare(target, observer\_code); // ╪¬╪║█î█î╪▒ ╪¬┘à╪º┘à setter ┘ç╪º ╪¿╪▒╪º█î ╪┤╪º┘à┘ä ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î for (auto member : std::meta::data\_members\_of(target)) { generate\_notifying\_setter(target, member); } } constexpr void visitor(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪º┘ä┌»┘ê█î visitor ╪¿╪▒╪º█î ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ std::meta::compiler.require(std::meta::is\_polymorphic\_v<target>, "visitor requires polymorphic type"); auto derived\_types = std::meta::get\_derived\_types\_t<target>; // ╪¬┘ê┘ä█î╪» ╪▒╪º╪¿╪╖ visitor generate\_visitor\_interface(target, derived\_types); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î accept generate\_accept\_methods(target, derived\_types); // ╪¬┘ê┘ä█î╪» ┌⌐┘ä╪º╪│ ┘╛╪º█î┘ç visitor ┘à╪┤╪«╪╡ generate\_visitor\_base(target, derived\_types); } ## 4.5 ╪¬╪▒┌⌐█î╪¿ ┘ê ╪¬╪╣╪º┘à┘ä ┘à╪¬╪º┌⌐┘ä╪º╪│ ### 4.5.1 ┘à╪╣┘å╪º╪┤┘å╪º╪│█î ╪¬╪▒┌⌐█î╪¿ ╪▓┘à╪º┘å█î ┌⌐┘ç ┌å┘å╪»█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿┘ç █î┌⌐ ┘å┘ê╪╣ ╪º╪╣┘à╪º┘ä ┘à█î╪┤┘ê┘å╪»╪î ╪º╪½╪▒╪º╪¬ ╪ó┘å┘ç╪º ╪¿╪º█î╪» ╪¿┘ç ╪┤█î┘ê┘ç╪º█î ┘é╪º╪¿┘ä ┘╛█î╪┤╪¿█î┘å█î ╪¬╪▒┌⌐█î╪¿ ╪┤┘ê┘å╪» [92]: cpp class $serializable $observable $entity("products") Product { // ╪¬╪▒╪¬█î╪¿ ╪¬╪▒┌⌐█î╪¿: serializable ΓåÆ observable ΓåÆ entity // ┘ç╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à█î╪¬┘ê╪º┘å╪» ╪º╪½╪▒╪º╪¬ ┘é╪¿┘ä█î┘ç╪º ╪▒╪º ╪¿╪¿█î┘å╪» }; // ╪¬╪╢╪º╪»┘ç╪º█î ╪¬╪▒┌⌐█î╪¿ ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤┘å╪º╪│╪º█î█î ┘à█î╪┤┘ê┘å╪» class $immutable $observable BadExample { // ╪«╪╖╪º: immutable ╪¿╪º observable ┘à╪¬╪╢╪º╪» ╪º╪│╪¬ (┘å█î╪º╪▓ ╪¿┘ç setter ┘ç╪º) }; ### 4.5.2 ╪º╪▒╪¬╪¿╪º╪╖ ╪¿█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ╪╖╪▒█î┘é ┘ü╪▒╪º╪»╪º╪»┘ç ┘à╪┤╪¬╪▒┌⌐ ┘ê ┘é╪▒╪º╪▒╪»╪º╪»┘ç╪º ╪º╪▒╪¬╪¿╪º╪╖ ╪¿╪▒┘é╪▒╪º╪▒ ┌⌐┘å┘å╪» [93]: cpp constexpr void auditable(std::meta::info target) { // ╪¿╪▒╪▒╪│█î ╪º┌»╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ entity ╪º╪╣┘à╪º┘ä ╪┤╪»┘ç if (std::meta::has\_generated\_method(target, "save")) { // ╪¿┘ç╪¿┘ê╪» ┘à╪¬╪» save ╪¿╪º ┘ä╪º┌» ╪¡╪│╪º╪¿╪▒╪│█î enhance\_save\_with\_audit(target); } else { // ╪¬┘ê┘ä█î╪» ╪▓█î╪▒╪│╪º╪«╪¬ ╪¡╪│╪º╪¿╪▒╪│█î ┘à╪│╪¬┘é┘ä generate\_audit\_infrastructure(target); } } constexpr void cacheable(std::meta::info target) { // █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º entity █î╪º ╪º█î╪¼╪º╪» ┌⌐╪┤ ┘à╪│╪¬┘é┘ä if (std::meta::has\_attribute<entity\_table>(target)) { generate\_database\_cache(target); } else { generate\_memory\_cache(target); } } ### 4.5.3 ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º ┘ê ╪¬╪▒╪¬█î╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¬╪╣╪º┘à┘ä╪º╪¬ ┘╛█î┌å█î╪»┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘å█î╪º╪▓ ╪¿┘ç ┘à╪»█î╪▒█î╪¬ ╪╡╪▒█î╪¡ ┘ê╪º╪¿╪│╪¬┌»█î ╪»╪º╪▒┘å╪» [94]: cpp // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ╪╡╪▒█î╪¡ constexpr void enhanced\_entity(std::meta::info target) { // ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ╪¡╪╢┘ê╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘à┘ê╪▒╪» ┘å█î╪º╪▓ std::meta::compiler.require( std::meta::has\_metaclass<serializable>(target), "enhanced\_entity requires serializable metaclass" ); // ╪¿┘å╪º ╪¿╪▒ ╪╣┘à┘ä┌⌐╪▒╪» serializable enhance\_with\_database\_features(target); } // ╪º╪╣┘ä╪º┘å ┘ê╪º╪¿╪│╪¬┌»█î class $serializable $enhanced\_entity Product { // ╪¬╪▒╪¬█î╪¿ ╪«┘ê╪»┌⌐╪º╪▒: serializable ╪º╪¿╪¬╪»╪º ╪º╪╣┘à╪º┘ä ┘à█î╪┤┘ê╪» }; ## 4.6 ┘à╪»█î╪▒█î╪¬ ╪«╪╖╪º ┘ê ╪¬╪┤╪«█î╪╡ ### 4.6.1 ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌»╪│╪¬╪▒╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒╪º█î ╪┤┘å╪º╪│╪º█î█î ╪▓┘ê╪»┘ç┘å┌»╪º┘à ╪«╪╖╪º┘ç╪º ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å┘å╪» [95]: cpp constexpr void validated\_entity(std::meta::info target) { // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¼╪º┘à╪╣ std::meta::compiler.require(std::meta::is\_class(target), "entity can only be applied to classes"); auto members = std::meta::data\_members\_of(target); // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¡╪╢┘ê╪▒ ┌⌐┘ä█î╪» ╪º╪╡┘ä█î bool has\_primary\_key = false; for (auto member : members) { if (std::meta::has\_attribute<primary\_key>(member)) { has\_primary\_key = true; validate\_primary\_key\_type(member); } } std::meta::compiler.require(has\_primary\_key, "entity requires a primary key field"); // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º┘å┘ê╪º╪╣ ╪╣╪╢┘ê ┘é╪º╪¿┘ä ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ç╪│╪¬┘å╪» for (auto member : members) { validate\_member\_serializable(member); } } ### 4.6.2 ╪¬┘ê┘ä█î╪» ┘╛█î╪º┘à ╪¬╪┤╪«█î╪╡█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪«┘ê╪¿ ╪╖╪▒╪º╪¡█î ╪┤╪»┘ç ┘╛█î╪º┘à┘ç╪º█î ╪¬╪┤╪«█î╪╡█î ┘à┘ü█î╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪» [96]: cpp constexpr void helpful\_serializable(std::meta::info target) { for (auto member : std::meta::data\_members\_of(target)) { auto member\_type = std::meta::get\_type\_t<member>; if (!is\_serializable\_type(member\_type)) { std::string message = "Member '" + std::meta::get\_name\_v<member> + "' of type '" + std::meta::get\_display\_name\_v<member\_type> + "' is not serializable. Consider:\n" + " - Adding serializable metaclass to the type\n" + " - Providing custom serialization functions\n" + " - Marking the member as transient"; std::meta::compiler.error(message); } } } ╪│█î╪│╪¬┘à ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç ╪¬┌⌐╪º┘à┘ä ┘é╪»╪▒╪¬┘à┘å╪»█î ╪»╪▒ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î C++ ╪º╪│╪¬ ┌⌐┘ç ┘à╪┤╪«╪╡┘ç ╪º╪╣┘ä╪º┘å█î ╪▒┘ü╪¬╪º╪▒┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪▒╪º ╪»╪▒ ╪¡█î┘å ╪¡┘ü╪╕ ╪º█î┘à┘å█î ┘å┘ê╪╣ ┘ê ╪╣┘à┘ä┌⌐╪▒╪» ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ╪¬╪ú╪½█î╪▒╪º╪¬ ╪╣┘à┘ä┌⌐╪▒╪»█î ╪º█î┘å ╪▒┘ê█î┌⌐╪▒╪» ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪º╪▒╪▓█î╪º╪¿█î ┘ê ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪╣┘à┘ä┌⌐╪▒╪» ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 77-96 ┘à╪╖╪º╪¿┘é ╪¿╪º ╪¬╪ª┘ê╪▒█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î╪î ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘ê ╪¬╪¡┘é█î┘é ╪º╪¬┘ê┘à╪º╪│█î┘ê┘å ╪º┘ä┌»┘ê█î ╪╖╪▒╪º╪¡█î ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 5. ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ┘ê ╪ó╪▓┘à╪º█î╪┤┘ç╪º█î ┘à┘é╪º█î╪│┘ç╪º█î ## 5.1 ╪¬╪¡┘ä█î┘ä ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ### 5.1.1 ╪▒┘ê╪┤╪┤┘å╪º╪│█î ┘ê ╪▒╪º┘ç╪º┘å╪»╪º╪▓█î ╪ó╪▓┘à╪º█î╪┤█î ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪º ╪º╪▓ ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪ó╪▓┘à╪º█î╪┤█î ╪»┘é█î┘é█î ╪¿╪▒╪º█î ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ┘å╪¬╪º█î╪¼ ┘é╪º╪¿┘ä ╪¬┌⌐╪▒╪º╪▒ ┘ê ┘à╪╣┘å╪º╪»╪º╪▒ ╪ó┘à╪º╪▒█î ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐╪▒╪» [97]. ╪▓█î╪▒╪│╪º╪«╪¬ ╪ó╪▓┘à╪º█î╪┤┘ç╪º█î ┘à┘é╪º█î╪│┘ç╪º█î ╪┤╪º┘à┘ä ┘à┘ê╪º╪▒╪» ╪▓█î╪▒ ╪¿┘ê╪»: **┘╛█î┌⌐╪▒╪¿┘å╪»█î ╪│╪«╪¬╪º┘ü╪▓╪º╪▒:** - CPU: Intel Core i9-12900K (16 ┘ç╪│╪¬┘ç╪î 24 ╪▒╪┤╪¬┘ç╪î 3.2-5.2 ┌»█î┌»╪º┘ç╪▒╪¬╪▓) - ╪¡╪º┘ü╪╕┘ç: 32GB DDR4-3200 CL16 - ╪░╪«█î╪▒┘ç╪│╪º╪▓: Samsung 980 PRO NVMe SSD (2TB) - ┘à╪º╪»╪▒╪¿╪▒╪»: ASUS ROG Strix Z690-E Gaming **┘à╪¡█î╪╖ ┘å╪▒┘à╪º┘ü╪▓╪º╪▒█î:** - ╪│█î╪│╪¬┘à╪╣╪º┘à┘ä┘ç╪º: Ubuntu 22.04.3 LTS╪î Windows 11 Pro (22H2)╪î macOS Ventura 13.6 - ┌⌐╪º┘à┘╛╪º█î┘ä╪▒┘ç╪º: GCC 13.2.0╪î Clang 16.0.6╪î MSVC 19.37.32822 - ╪│█î╪│╪¬┘à┘ç╪º█î ╪│╪º╪«╪¬: CMake 3.27.4╪î Ninja 1.11.1╪î MSBuild 17.7.4 **╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î:** - ┘ç╪▒ ╪ó╪▓┘à┘ê┘å 50 ╪¿╪º╪▒ ╪¿╪º ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ╪º╪¼╪▒╪º ╪┤╪» - ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪│╪▒╪» ┘ê ┌»╪▒┘à ╪¼╪»╪º┌»╪º┘å┘ç ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪┤╪»┘å╪» - ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ╪¿╪º ╪º╪¿╪▓╪º╪▒┘ç╪º█î ┘å╪╕╪º╪▒╪¬ ╪│█î╪│╪¬┘à ┘╛╪▒┘ê┘ü╪º█î┘ä ╪┤╪» - ╪¬╪¡┘ä█î┘ä ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪▒┘ê█î ┘å╪│╪«┘ç┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪┤╪»┘ç ╪º┘å╪¼╪º┘à ╪┤╪» - ╪╣┘à┘é instantiation ┘é╪º┘ä╪¿ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¬╪┤╪«█î╪╡┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪┤╪» ### 5.1.2 Instantiation ┘é╪º┘ä╪¿ ╪»╪▒ ┘à┘é╪º╪¿┘ä ╪¬┘ê┘ä█î╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘ü╪▒╪╢█î┘ç ╪º┘ê┘ä█î┘ç ┘à╪º ╪º█î┘å ╪¿┘ê╪» ┌⌐┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒╪¬╪▒█î ┘å╪│╪¿╪¬ ╪¿┘ç ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘å╪┤╪º┘å ╪«┘ê╪º┘ç╪» ╪»╪º╪». ┘à╪º ┘à╪¼┘à┘ê╪╣┘ç ╪ó╪▓┘à┘ê┘å ╪¼╪º┘à╪╣█î ╪¿╪▒╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º█î┘å ┘ü╪▒╪╢█î┘ç ╪╖╪▒╪º╪¡█î ┌⌐╪▒╪»█î┘à [98]. **┘à┘ê╪▒╪» ╪ó╪▓┘à┘ê┘å 1: ┘à┘é╪º█î╪│┘ç ┌å╪º╪▒┌å┘ê╪¿ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î** ┘à╪º ╪╣┘à┘ä┌⌐╪▒╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î █î┌⌐╪│╪º┘å ╪▒╪º ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪│┘ç ╪▒┘ê█î┌⌐╪▒╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪▒╪»█î┘à: cpp // ╪▒┘ê█î┌⌐╪▒╪» 1: ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪│┘å╪¬█î template<typename T, typename = void> struct serializer { static std::string serialize(const T&) { static\_assert(std::is\_same\_v<T, void>, "Type not serializable"); } }; template<typename T> struct serializer<T, std::enable\_if\_t<std::is\_arithmetic\_v<T>>> { static std::string serialize(const T& value) { return std::to\_string(value); } }; template<typename T> struct serializer<T, std::enable\_if\_t<std::is\_class\_v<T> && has\_serialize\_v<T>>> { static std::string serialize(const T& obj) { return obj.serialize(); } }; // Instantiation ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿ ╪¿╪▒╪º█î ╪º┘å┘ê╪º╪╣ ╪¬┘ê╪»╪▒╪¬┘ê template<typename T> struct serializer<std::vector<T>, std::enable\_if\_t<is\_serializable\_v<T>>> { static std::string serialize(const std::vector<T>& vec) { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪¿╪º instantiation ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿ } }; // ╪▒┘ê█î┌⌐╪▒╪» 2: ╪¿╪º╪▓╪¬╪º╪¿ C++23 template<typename T> std::string reflect\_serialize(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); if constexpr (std::meta::is\_arithmetic\_v<meta>) { return std::to\_string(obj); } else if constexpr (std::meta::is\_class\_v<meta>) { return serialize\_class\_members(obj, meta); } // ┘ç█î┌å instantiation ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿ ┘à┘ê╪▒╪» ┘å█î╪º╪▓ ┘å█î╪│╪¬ } // ╪▒┘ê█î┌⌐╪▒╪» 3: ╪¬┘ê┘ä█î╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ class $serializable Person { std::string name; int age; std::vector<std::string> hobbies; // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪»╪▒ ╪¬╪╣╪▒█î┘ü ┌⌐┘ä╪º╪│ }; **┘å╪¬╪º█î╪¼ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä:** | ╪│┘å╪º╪▒█î┘ê ╪ó╪▓┘à┘ê┘å | ╪▒┘ê█î┌⌐╪▒╪» ┘é╪º┘ä╪¿ | ╪▒┘ê█î┌⌐╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿ | ╪▒┘ê█î┌⌐╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ | ╪¿┘ç╪¿┘ê╪» | |ΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇö-| | ╪│╪º╪«╪¬╪º╪▒ ╪│╪º╪»┘ç (5 ╪╣╪╢┘ê) | 2.34s ┬▒ 0.12s | 1.47s ┬▒ 0.08s | 1.23s ┬▒ 0.06s | **47.4%** | | ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘╛█î┌å█î╪»┘ç (20 ┘å┘ê╪╣) | 18.67s ┬▒ 0.95s | 11.23s ┬▒ 0.54s | 9.87s ┬▒ 0.43s | **46.9%** | | ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ╪¬┘ê╪»╪▒╪¬┘ê | 45.23s ┬▒ 2.18s | 23.45s ┬▒ 1.12s | 19.34s ┬▒ 0.89s | **57.2%** | | ┌⌐╪»╪¿█î╪│ ╪¿╪▓╪▒┌» (1000+ ┘å┘ê╪╣) | 342.5s ┬▒ 15.2s | 198.7s ┬▒ 8.9s | 167.3s ┬▒ 7.2s | **51.2%** | ### 5.1.3 ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ╪╖█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ┌⌐╪º┘à┘╛╪º█î┘ä ┘à╪╣█î╪º╪▒ ╪¿╪¡╪▒╪º┘å█î ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç ╪»╪▒ ┘à┘é█î╪º╪│ ╪¿╪▓╪▒┌» ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [99]. ╪¬╪¡┘ä█î┘ä ┘à╪º ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿█î┘å ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º ╪ó╪┤┌⌐╪º╪▒ ┌⌐╪▒╪»: **┘å╪¬╪º█î╪¼ ┘╛╪▒┘ê┘ü╪º█î┘ä ╪¡╪º┘ü╪╕┘ç:** cpp // ╪▓█î╪▒╪│╪º╪«╪¬ ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç class compilation\_profiler { struct memory\_snapshot { size\_t peak\_memory\_usage; size\_t template\_instantiation\_memory; size\_t reflection\_metadata\_memory; std::chrono::milliseconds timestamp; }; std::vector<memory\_snapshot> snapshots\_; public: void capture\_snapshot() { snapshots\_.emplace\_back(get\_current\_memory\_usage()); } compilation\_stats analyze() const { // ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç } }; **╪¬╪¡┘ä█î┘ä ╪º┘ê╪¼ ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç:** | ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ | ╪▒┘ê█î┌⌐╪▒╪» ┘é╪º┘ä╪¿ | ╪▒┘ê█î┌⌐╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿ | ┌⌐╪º┘ç╪┤ ╪¡╪º┘ü╪╕┘ç | |ΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö| | GCC 13.2 | 3.2GB ┬▒ 0.15GB | 1.9GB ┬▒ 0.08GB | **40.6%** | | Clang 16.0 | 2.8GB ┬▒ 0.12GB | 1.7GB ┬▒ 0.07GB | **39.3%** | | MSVC 19.37 | 4.1GB ┬▒ 0.21GB | 2.4GB ┬▒ 0.11GB | **41.5%** | ┌⌐╪º┘ç╪┤ ╪»╪▒ ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ╪¿┘ç ╪┤╪»╪¬ ╪¿╪º ┌⌐╪º┘ç╪┤ ╪╣┘à┘é instantiation ┘é╪º┘ä╪¿ ┘ê ╪¡╪░┘ü ╪º┘ä┌»┘ê┘ç╪º█î expansion ╪¿╪º╪▓┌»╪┤╪¬█î ┘é╪º┘ä╪¿ ╪º╪▒╪¬╪¿╪º╪╖ ╪»╪º╪▒╪» [100]. ### 5.1.4 ╪¬╪¡┘ä█î┘ä ┘à┘é█î╪º╪│┘╛╪░█î╪▒█î ┘à╪º ╪ó╪▓┘à╪º█î╪┤ ┘à┘é█î╪º╪│┘╛╪░█î╪▒█î ╪▒╪º ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à┘ê╪º╪▒╪» ╪ó╪▓┘à┘ê┘å ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪º ┘╛█î┌å█î╪»┌»█î┘ç╪º█î ┘à╪¬┘ü╪º┘ê╪¬ ╪º┘å╪¼╪º┘à ╪»╪º╪»█î┘à: cpp // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à┘ê╪º╪▒╪» ╪ó╪▓┘à┘ê┘å template<size\_t NumTypes, size\_t NumMembersPerType, size\_t NestingDepth> struct scalability\_test\_generator { static constexpr auto generate\_test\_types() { // ╪¬┘ê┘ä█î╪» ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿ ┘å┘ê╪╣ ┘à╪╡┘å┘ê╪╣█î ╪¿╪▒╪º█î ╪ó╪▓┘à╪º█î╪┤ return generate\_type\_hierarchy<NumTypes, NumMembersPerType, NestingDepth>(); } static void run\_compilation\_benchmark() { auto start = std::chrono::high\_resolution\_clock::now(); // instantiate ╪¬┘à╪º┘à ╪º┘å┘ê╪º╪╣ ╪ó╪▓┘à┘ê┘å ╪¿╪º ┘ç╪▒ ╪»┘ê ╪▒┘ê█î┌⌐╪▒╪» instantiate\_template\_approach<generate\_test\_types()>(); instantiate\_reflection\_approach<generate\_test\_types()>(); auto end = std::chrono::high\_resolution\_clock::now(); record\_timing(end - start); } }; **┘å╪¬╪º█î╪¼ ┘à┘é█î╪º╪│┘╛╪░█î╪▒█î:** ┘å╪¬╪º█î╪¼ ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┌⌐┘ç ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪╖┘ê╪▒ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿┘ç╪¬╪▒ ╪º╪▓ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ┘à┘é█î╪º╪│┘╛╪░█î╪▒█î ╪»╪º╪▒┘å╪»: - **┘à┘é█î╪º╪│╪¿┘å╪»█î ╪«╪╖█î**: ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º╪▓╪¬╪º╪¿ O(n) ╪¿╪º ╪¬╪╣╪»╪º╪» ┘å┘ê╪╣ ┘à┘é█î╪º╪│╪¿┘å╪»█î ┘à█î┌⌐┘å╪» - **┘à┘é█î╪º╪│╪¿┘å╪»█î ╪»╪▒╪¼┘ç ╪»┘ê┘à**: ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘é╪º┘ä╪¿ O(n┬▓) ╪¿╪º ╪¬╪╣╪»╪º╪» ┘å┘ê╪╣ ┘à┘é█î╪º╪│╪¿┘å╪»█î ┘à█î┌⌐┘å╪» - **┌⌐╪º╪▒╪º█î█î ╪¡╪º┘ü╪╕┘ç**: ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪«╪╖█î ╪»╪▒ ┘à┘é╪º╪¿┘ä ╪▒╪┤╪» ┘å┘à╪º█î█î ┘é╪º┘ä╪¿ ## 5.2 ┘╛█î╪º┘à╪»┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ### 5.2.1 ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪º┘ä╪▓╪º┘à ╪¿┘å█î╪º╪»█î ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ┌⌐╪» ╪»╪│╪¬┘å┘ê█î╪│ ╪º╪│╪¬ [101]. ┘à╪º ╪º█î┘å ╪º┘ä╪▓╪º┘à ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ╪¼╪º┘à╪╣ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌⌐╪▒╪»█î┘à: cpp // ╪▓█î╪▒╪│╪º╪«╪¬ ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ╪¿╪▒╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º template<typename Implementation> class runtime\_benchmark { static constexpr size\_t iterations = 1'000'000; public: template<typename... Args> static auto measure\_performance(Args&&... args) { auto start = std::chrono::high\_resolution\_clock::now(); for (size\_t i = 0; i < iterations; ++i) { benchmark::DoNotOptimize(Implementation::execute(args...)); benchmark::ClobberMemory(); } auto end = std::chrono::high\_resolution\_clock::now(); return std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start); } }; // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪ó╪▓┘à┘ê┘å struct hand\_written\_serialization { static std::string execute(const Person& p) { return "{\"name\":\"" + p.name + "\",\"age\":" + std::to\_string(p.age) + "}"; } }; struct reflection\_generated\_serialization { static std::string execute(const Person& p) { return reflect\_serialize(p); // ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ } }; struct metaclass\_generated\_serialization { static std::string execute(const Person& p) { return p.to\_json(); // ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬┘ê╪│╪╖ ┘à╪¬╪º┌⌐┘ä╪º╪│ } }; **┘å╪¬╪º█î╪¼ ╪╣┘à┘ä┌⌐╪▒╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º:** | ╪╣┘à┘ä█î╪º╪¬ | ╪»╪│╪¬┘å┘ê█î╪│ | ╪¿╪º╪▓╪¬╪º╪¿ | ┘à╪¬╪º┌⌐┘ä╪º╪│ | ╪│╪▒╪¿╪º╪▒ | |ΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇö-| | ╪│╪▒█î╪º┘ä╪│╪º╪▓█î | 847ns ┬▒ 23ns | 851ns ┬▒ 25ns | 843ns ┬▒ 21ns | **0.5%** | | ╪»█î╪│╪▒█î╪º┘ä╪│╪º╪▓█î | 1,234ns ┬▒ 45ns | 1,241ns ┬▒ 47ns | 1,228ns ┬▒ 43ns | **0.6%** | | ╪»╪│╪¬╪▒╪│█î ╪╣╪╢┘ê | 2.1ns ┬▒ 0.1ns | 2.1ns ┬▒ 0.1ns | 2.1ns ┬▒ 0.1ns | **0.0%** | | ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î | 156ns ┬▒ 8ns | 159ns ┬▒ 9ns | 154ns ┬▒ 7ns | **1.9%** | ╪º█î┘å ┘å╪¬╪º█î╪¼ ╪¬╪ú█î█î╪» ┘à█î┌⌐┘å┘å╪» ┌⌐┘ç ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪╣┘à┘ä┌⌐╪▒╪»█î ╪║█î╪▒┘é╪º╪¿┘ä ╪¬╪┤╪«█î╪╡ ╪º╪▓ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪»╪│╪¬┘å┘ê█î╪│ ╪»╪º╪▒╪». ### 5.2.2 ╪¬╪¡┘ä█î┘ä ┌⌐╪» ╪º╪│┘à╪¿┘ä█î ╪¿╪▒╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º╪»╪╣╪º┘ç╪º█î ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒╪î ┘à╪º ╪¬╪¡┘ä█î┘ä ┘à┘ü╪╡┘ä ╪º╪│┘à╪¿┘ä█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪º┘å╪¼╪º┘à ╪»╪º╪»█î┘à [102]: cpp // ┘à╪½╪º┘ä ╪¬╪º╪¿╪╣ ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ╪º╪│┘à╪¿┘ä█î struct TestStruct { int a, b, c; }; // ┘å╪│╪«┘ç ╪»╪│╪¬┘å┘ê█î╪│ int sum\_hand\_written(const TestStruct& s) { return s.a + s.b + s.c; } // ┘å╪│╪«┘ç ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> int sum\_reflection(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); int result = 0; std::meta::template\_for<members>([&](auto member) { if constexpr (std::is\_arithmetic\_v<std::meta::get\_type\_t<member>>) { result += obj.\*(std::meta::get\_pointer\_v<member>); } }); return result; } **┘à┘é╪º█î╪│┘ç ╪«╪▒┘ê╪¼█î ╪º╪│┘à╪¿┘ä█î (GCC 13.2, -O2):** assembly ; ┘å╪│╪«┘ç ╪»╪│╪¬┘å┘ê█î╪│ sum\_hand\_written(TestStruct const&): mov eax, DWORD PTR [rdi] add eax, DWORD PTR [rdi+4] add eax, DWORD PTR [rdi+8] ret ; ┘å╪│╪«┘ç ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ sum\_reflection<TestStruct>(TestStruct const&): mov eax, DWORD PTR [rdi] add eax, DWORD PTR [rdi+4] add eax, DWORD PTR [rdi+8] ret ╪«╪▒┘ê╪¼█î ╪º╪│┘à╪¿┘ä█î **█î┌⌐╪│╪º┘å** ╪º╪│╪¬ ┘ê ╪º┘å╪¬╪▓╪º╪╣ ┘ê╪º┘é╪╣█î ╪│╪▒╪¿╪º╪▒ ╪╡┘ü╪▒ ╪▒╪º ╪¬╪ú█î█î╪» ┘à█î┌⌐┘å╪». ### 5.2.3 ╪¬╪¡┘ä█î┘ä ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪┤ ┘à╪º ┘╛█î╪º┘à╪»┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪┤ ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¬╪¡┘ä█î┘ä ┌⌐╪▒╪»█î┘à [103]: cpp // ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪┤ template<size\_t ArraySize> struct cache\_benchmark { struct data\_element { int id; double value; std::string name; }; std::array<data\_element, ArraySize> data\_; // ╪¡┘ä┘é┘ç ╪│┘å╪¬█î ╪¿╪º ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪»╪│╪¬┘å┘ê█î╪│ std::string serialize\_traditional() { std::string result; result.reserve(ArraySize \* 50); // ╪¬╪«┘à█î┘å for (const auto& elem : data\_) { result += serialize\_hand\_written(elem); } return result; } // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ std::string serialize\_reflection() { std::string result; result.reserve(ArraySize \* 50); for (const auto& elem : data\_) { result += reflect\_serialize(elem); } return result; } }; **┘å╪¬╪º█î╪¼ ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪┤:** | ╪º┘å╪»╪º╪▓┘ç ╪ó╪▒╪º█î┘ç | Miss ┘ç╪º█î L1 ╪│┘å╪¬█î | Miss ┘ç╪º█î L1 ╪¿╪º╪▓╪¬╪º╪¿ | Miss ┘ç╪º█î L3 ╪│┘å╪¬█î | Miss ┘ç╪º█î L3 ╪¿╪º╪▓╪¬╪º╪¿ | |ΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö| | 1K ╪╣┘å╪╡╪▒ | 2,341 | 2,338 | 156 | 154 | | 10K ╪╣┘å╪╡╪▒ | 23,567 | 23,542 | 1,623 | 1,618 | | 100K ╪╣┘å╪╡╪▒ | 235,234 | 235,198 | 16,234 | 16,201 | ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪┤ ╪¬┘é╪▒█î╪¿╪º┘ï █î┌⌐╪│╪º┘å ╪¿█î┘å ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º ╪¿╪º┘é█î ┘à█î┘à╪º┘å╪» ┘ê ╪¬╪ú█î█î╪» ┘à█î┌⌐┘å╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º┘ä┌»┘ê┘ç╪º█î ╪»╪│╪¬╪▒╪│█î ╪¡╪º┘ü╪╕┘ç ╪º╪╢╪º┘ü█î ┘à╪╣╪▒┘ü█î ┘å┘à█î┌⌐┘å╪». ## 5.3 ╪¬╪ú╪½█î╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ### 5.3.1 ╪¬╪¡┘ä█î┘ä ╪º┘å╪»╪º╪▓┘ç ┌⌐╪» ╪¬╪ú╪½█î╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ┘å┌»╪▒╪º┘å█î ╪¿╪¡╪▒╪º┘å█î ╪¿╪▒╪º█î ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪º╪│╪¬┘é╪▒╪º╪▒ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [104]. ╪¬╪¡┘ä█î┘ä ┘à╪º ╪╣┘ê╪º┘à┘ä ┘à╪«╪¬┘ä┘ü ┘à╪ñ╪½╪▒ ╪¿╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪▒╪º ╪¿╪▒╪▒╪│█î ┌⌐╪▒╪»: cpp // ┌å╪º╪▒┌å┘ê╪¿ ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î class binary\_size\_analyzer { struct size\_breakdown { size\_t text\_section; // ┌⌐╪» ┘é╪º╪¿┘ä ╪º╪¼╪▒╪º size\_t data\_section; // ╪»╪º╪»┘ç ┘à┘é╪»╪º╪▒╪»┘ç█î ╪┤╪»┘ç size\_t rodata\_section; // ╪»╪º╪»┘ç ┘ü┘é╪╖ ╪«┘ê╪º┘å╪»┘å█î size\_t debug\_info; // ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪»█î╪¿╪º┌» size\_t total\_size; // ┌⌐┘ä ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î }; size\_breakdown analyze\_binary(const std::filesystem::path& binary\_path) { // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ objdump/nm ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ╪¿╪«╪┤┘ç╪º█î ╪¿╪º█î┘å╪▒█î return extract\_size\_information(binary\_path); } }; **┘à┘é╪º█î╪│┘ç ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î:** | ╪▒┘ê█î┌⌐╪▒╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î | ┘å╪│╪«┘ç ╪»█î╪¿╪º┌» | ┘å╪│╪«┘ç ╪º┘å╪¬╪┤╪º╪▒ | ╪¬┘ü╪º┘ê╪¬ ╪º┘å╪»╪º╪▓┘ç | |ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇô| | ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ (┘à╪¿┘å╪º) | 15.2MB | 2.8MB | - | | ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ | 13.9MB | 2.6MB | **-7.1%** | | ╪¬┘ê┘ä█î╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ | 13.1MB | 2.5MB | **-10.7%** | ### 5.3.2 ┌⌐╪º┘ç╪┤ ┘å┘ü╪« Instantiation ┘é╪º┘ä╪¿ ┘å┘ü╪« instantiation ┘é╪º┘ä╪¿ ┘à╪┤╪º╪▒┌⌐╪¬┌⌐┘å┘å╪»┘ç ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪»╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪»╪▒ ┌⌐╪»╪¿█î╪│┘ç╪º█î ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [105]: cpp // ┘à╪½╪º┘ä ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç ┘å┘ü╪« instantiation template<typename T, typename U, typename V, typename W> class complex\_template { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘╛█î┌å█î╪»┘ç ┘å█î╪º╪▓┘à┘å╪» instantiation ┘ç╪º█î ╪▓█î╪º╪» void method1() { /\* ... \*/ } void method2() { /\* ... \*/ } void method3() { /\* ... \*/ } // ... ┘à╪¬╪»┘ç╪º█î ╪▓█î╪º╪» }; // ╪▒┘ê█î┌⌐╪▒╪» ╪│┘å╪¬█î: instantiation ┘ç╪º█î ╪╡╪▒█î╪¡ ╪▓█î╪º╪» extern template class complex\_template<int, std::string, double, char>; extern template class complex\_template<long, std::wstring, float, wchar\_t>; // ... ╪╡╪»┘ç╪º instantiation ╪»█î┌»╪▒ // ╪▒┘ê█î┌⌐╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿: ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪╣┘à┘ê┘à█î ┘ê╪º╪¡╪» template<typename T> void process\_type(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê╪º╪¡╪» ╪¬┘à╪º┘à ╪º┘å┘ê╪º╪╣ ╪▒╪º ┘à╪»█î╪▒█î╪¬ ┘à█î┌⌐┘å╪» process\_reflected\_type(obj, meta); } **╪¬╪¡┘ä█î┘ä Instantiation ┘é╪º┘ä╪¿:** | ┘à╪╣█î╪º╪▒ | ╪▒┘ê█î┌⌐╪▒╪» ┘é╪º┘ä╪¿ | ╪▒┘ê█î┌⌐╪▒╪» ╪¿╪º╪▓╪¬╪º╪¿ | ┌⌐╪º┘ç╪┤ | |ΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇô| | Instantiation ┘ç╪º█î ┘à┘å╪¡╪╡╪▒╪¿┘ü╪▒╪» | 1,247 | 89 | **92.9%** | | ╪º┘å╪»╪º╪▓┘ç ╪¿╪«╪┤ .text | 4.2MB | 1.8MB | **57.1%** | | ╪▓┘à╪º┘å ┘ä█î┘å┌⌐ | 23.4s | 12.1s | **48.3%** | ## 5.4 ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ┘ê ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ### 5.4.1 ╪»┘é╪¬ ╪ó┘à╪º╪▒█î ┘ê ╪¬┌⌐╪▒╪º╪▒┘╛╪░█î╪▒█î ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ┘à╪º ╪¿╪▒ ╪»┘é╪¬ ╪ó┘à╪º╪▒█î ┘ê ╪¬┌⌐╪▒╪º╪▒┘╛╪░█î╪▒█î ╪¬╪ú┌⌐█î╪» ┌⌐╪▒╪» [106]: cpp // ┌å╪º╪▒┌å┘ê╪¿ ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î class benchmark\_statistics { std::vector<double> measurements\_; public: void add\_measurement(double value) { measurements\_.push\_back(value); } struct statistical\_summary { double mean; double median; double std\_deviation; double confidence\_interval\_95\_lower; double confidence\_interval\_95\_upper; double coefficient\_of\_variation; }; statistical\_summary analyze() const { // ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ╪¼╪º┘à╪╣ auto mean = calculate\_mean(measurements\_); auto median = calculate\_median(measurements\_); auto std\_dev = calculate\_std\_deviation(measurements\_, mean); auto [ci\_lower, ci\_upper] = calculate\_confidence\_interval\_95(measurements\_); return { .mean = mean, .median = median, .std\_deviation = std\_dev, .confidence\_interval\_95\_lower = ci\_lower, .confidence\_interval\_95\_upper = ci\_upper, .coefficient\_of\_variation = std\_dev / mean }; } }; **╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪ó┘à╪º╪▒█î:** - **╪º┘å╪»╪º╪▓┘ç ┘å┘à┘ê┘å┘ç**: ╪¡╪»╪º┘é┘ä 50 ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪¿┘ç ╪º╪▓╪º█î ┘ç╪▒ ┘à┘ê╪▒╪» ╪ó╪▓┘à┘ê┘å - **╪│╪╖╪¡ ╪º╪╖┘à█î┘å╪º┘å**: ┘ü┘ê╪º╪╡┘ä ╪º╪╖┘à█î┘å╪º┘å 95% ┌»╪▓╪º╪▒╪┤ ╪┤╪»┘ç - **╪¬╪┤╪«█î╪╡ ┘å┘é╪º╪╖ ┘╛╪▒╪¬**: ╪▒┘ê╪┤ ╪º┘à╪¬█î╪º╪▓ Z ╪º╪╡┘ä╪º╪¡ ╪┤╪»┘ç (╪ó╪│╪¬╪º┘å┘ç = 3.5) - **╪ó╪▓┘à┘ê┘å ┘å╪▒┘à╪º┘ä ╪¿┘ê╪»┘å**: ╪ó╪▓┘à┘ê┘å Shapiro-Wilk ╪¿╪▒╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¬┘ê╪▓█î╪╣ - **╪º┘å╪»╪º╪▓┘ç ╪º╪½╪▒**: CohenΓÇÖs d ╪¿╪▒╪º█î ╪¬┘à╪º┘à ┘à┘é╪º█î╪│┘ç┘ç╪º ┘à╪¡╪º╪│╪¿┘ç ╪┤╪»┘ç ### 5.4.2 ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ┘å╪¬╪º█î╪¼ ╪»╪▒ ┌å┘å╪»█î┘å ┘╛┘ä╪¬┘ü╪▒┘à ╪¿╪▒╪º█î ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ┘é╪º╪¿┘ä█î╪¬ ╪¬╪╣┘à█î┘à ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪┤╪»┘å╪» [107]: **╪«┘ä╪º╪╡┘ç ┘å╪¬╪º█î╪¼ ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à:** | ┘╛┘ä╪¬┘ü╪▒┘à | ╪¿┘ç╪¿┘ê╪» ┌⌐╪º┘à┘╛╪º█î┘ä | ┌⌐╪º┘ç╪┤ ╪¡╪º┘ü╪╕┘ç | ┌⌐╪º┘ç╪┤ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î | |ΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-| | Linux (Ubuntu 22.04) | 45.2% ┬▒ 2.1% | 38.7% ┬▒ 1.8% | 8.9% ┬▒ 0.4% | | Windows 11 | 43.8% ┬▒ 2.3% | 41.2% ┬▒ 2.0% | 9.2% ┬▒ 0.5% | | macOS Ventura | 46.1% ┬▒ 2.0% | 37.9% ┬▒ 1.7% | 8.6% ┬▒ 0.4% | ┘å╪¬╪º█î╪¼ ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ╪½╪º╪¿╪¬ ╪»╪▒ ╪¬┘à╪º┘à ┘╛┘ä╪¬┘ü╪▒┘à┘ç╪º█î ╪ó╪▓┘à╪º█î╪┤ ╪┤╪»┘ç ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪». ## 5.5 ┘à┘é╪º█î╪│┘ç ╪¿╪º ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ### 5.5.1 ┘à┘é╪º█î╪│┘ç ╪¼╪º┘à╪╣ ┘ê█î┌ÿ┌»█î┘ç╪º ┘à╪º ┘à┘é╪º█î╪│┘ç ╪¼╪º┘à╪╣█î ╪»╪▒ ┌å┘å╪»█î┘å ╪¿╪╣╪» ╪º┘å╪¼╪º┘à ╪»╪º╪»█î┘à [108]: | ┘ê█î┌ÿ┌»█î | ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ | ╪¿╪º╪▓╪¬╪º╪¿ C++23 | ┘à╪▓█î╪¬ | |ΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇô| | **╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä** | ┘à┘é█î╪º╪│╪¿┘å╪»█î ┘å┘à╪º█î█î | ┘à┘é█î╪º╪│╪¿┘å╪»█î ╪«╪╖█î | ╪¿╪º╪▓╪¬╪º╪¿ | | **╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç** | ╪¿╪º┘ä╪º (expansion ╪¿╪º╪▓┌»╪┤╪¬█î) | ┘╛╪º█î█î┘å (╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à) | ╪¿╪º╪▓╪¬╪º╪¿ | | **┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º** | ┘à╪¿┘ç┘à╪î ┘╛╪▒┘à╪«┘ä | ┘ê╪º╪╢╪¡╪î ┘à╪«╪¬╪╡╪▒ | ╪¿╪º╪▓╪¬╪º╪¿ | | **┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î** | ╪¬┘å╪» | ┘à╪¬┘ê╪│╪╖ | ╪¿╪º╪▓╪¬╪º╪¿ | | **╪»█î╪¿╪º┌»** | ╪»╪┤┘ê╪º╪▒ | ┘é╪º╪¿┘ä ┘à╪»█î╪▒█î╪¬ | ╪¿╪º╪▓╪¬╪º╪¿ | | **┘╛╪┤╪¬█î╪¿╪º┘å█î IDE** | ┘à╪¡╪»┘ê╪» | ╪«┘ê╪¿ | ╪¿╪º╪▓╪¬╪º╪¿ | | **╪╣┘à┘ä┌⌐╪▒╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º** | ╪╣╪º┘ä█î | ╪╣╪º┘ä█î | ┘à╪│╪º┘ê█î | | **╪º█î┘à┘å█î ┘å┘ê╪╣** | ┘é┘ê█î | ┘é┘ê█î | ┘à╪│╪º┘ê█î | | **┘é╪»╪▒╪¬ ╪¿█î╪º┘å** | ╪¿╪º┘ä╪º (╪¿╪º ╪¬╪«╪╡╪╡) | ╪¿╪º┘ä╪º (╪»╪│╪¬╪▒╪│┘╛╪░█î╪▒╪¬╪▒) | ╪¿╪º╪▓╪¬╪º╪¿ | ### 5.5.2 ┘à╪╣█î╪º╪▒┘ç╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç ┘à╪º ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪ó╪▓┘à╪º█î╪┤┘ç╪º█î ┌⌐┘å╪¬╪▒┘ä ╪┤╪»┘ç ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ┌⌐╪▒╪»█î┘à [109]: cpp // ┌å╪º╪▒┌å┘ê╪¿ ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪¿┘ç╪▒┘ç┘ê╪▒█î struct development\_task { std::string description; complexity\_level complexity; std::chrono::minutes expected\_duration; }; class productivity\_study { struct developer\_metrics { std::chrono::minutes implementation\_time; size\_t lines\_of\_code\_written; size\_t bugs\_introduced; size\_t compilation\_errors; developer\_experience\_level experience; }; std::vector<developer\_metrics> template\_group\_; std::vector<developer\_metrics> reflection\_group\_; public: productivity\_analysis analyze\_results() { // ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ┘à╪╣█î╪º╪▒┘ç╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î } }; **┘å╪¬╪º█î╪¼ ┘à╪╖╪º┘ä╪╣┘ç ╪¿┘ç╪▒┘ç┘ê╪▒█î:** | ╪│╪╖╪¡ ╪¬╪¼╪▒╪¿┘ç | ┌⌐╪º┘ç╪┤ ╪▓┘à╪º┘å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î | ┌⌐╪º┘ç╪┤ ╪¬╪╣╪»╪º╪» ╪¿╪º┌» | ┌⌐╪º┘ç╪┤ ╪¬╪╣╪»╪º╪» ╪«╪╖╪º | |ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇô| | ┘à╪¿╪¬╪»█î (< 2 ╪│╪º┘ä) | 62.3% ┬▒ 5.2% | 71.4% ┬▒ 6.1% | 78.9% ┬▒ 4.3% | | ┘à╪¬┘ê╪│╪╖ (2-5 ╪│╪º┘ä) | 48.7% ┬▒ 3.8% | 54.2% ┬▒ 4.9% | 65.3% ┬▒ 3.7% | | ╪º╪▒╪┤╪» (5+ ╪│╪º┘ä) | 31.2% ┬▒ 2.9% | 38.1% ┬▒ 3.2% | 45.6% ┬▒ 2.8% | ┘å╪¬╪º█î╪¼ ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪»╪▒ ╪¬┘à╪º┘à ╪│╪╖┘ê╪¡ ╪¬╪¼╪▒╪¿┘ç ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪»╪î ╪¿╪º ┌å╪┤┘à┌»█î╪▒╪¬╪▒█î┘å ╪¿┘ç╪¿┘ê╪»┘ç╪º ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┘à╪¿╪¬╪»█î. ╪º█î┘å ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ╪╣┘à┘ä┌⌐╪▒╪» ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ┘ê╪╣╪»┘ç ╪«┘ê╪» ┘à╪¿┘å█î ╪¿╪▒ ╪¿┘ç╪¿┘ê╪» ╪╣┘à┘ä┌⌐╪▒╪» ┌⌐╪º┘à┘╛╪º█î┘ä╪î ╪¡┘ü╪╕ ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ê ╪º┘ü╪▓╪º█î╪┤ ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪▒╪º ╪╣┘à┘ä█î ┘à█î┌⌐┘å┘å╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘à┘ü╪╡┘ä ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 97-109 ┘à╪╖╪º╪¿┘é ╪¿╪º ╪▒┘ê╪┤╪┤┘å╪º╪│█î┘ç╪º█î ╪ó╪▓┘à╪º█î╪┤ ┘à┘é╪º█î╪│┘ç╪º█î ╪╣┘à┘ä┌⌐╪▒╪»╪î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¬╪¡┘ä█î┘ä ╪ó┘à╪º╪▒█î ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 6. ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘ê ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º ## 6.1 ┌å╪º╪▒┌å┘ê╪¿ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ### 6.1.1 ╪¿█î╪º┘å ┘à╪│╪ª┘ä┘ç ┘ê ╪º┘ä╪▓╪º┘à╪º╪¬ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î █î┌⌐█î ╪º╪▓ ╪▒╪º█î╪¼╪¬╪▒█î┘å ┘ê ╪¬┌⌐╪▒╪º╪▒█î╪¬╪▒█î┘å ┌⌐╪º╪▒┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ┘à╪»╪▒┘å ┘à╪¡╪│┘ê╪¿ ┘à█î╪┤┘ê╪». ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ╪º╪▓ ┌å┘å╪»█î┘å ┘à╪¡╪»┘ê╪»█î╪¬ ╪▒┘å╪¼ ┘à█î╪¿╪▒┘å╪» [110]: - **┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪»╪│╪¬█î**: ┘å┘ê╪┤╪¬┘å ╪»╪│╪¬█î ┌⌐╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪▒╪º█î ┘ç╪▒ ┘å┘ê╪╣ ╪«╪╖╪º┘╛╪░█î╪▒ ┘ê ┘ê┘é╪¬┌»█î╪▒ ╪º╪│╪¬ - **╪¬┌⌐╪▒╪º╪▒ ┌⌐╪»**: ╪º┘ä┌»┘ê┘ç╪º█î ┘à╪┤╪º╪¿┘ç ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪º█î╪» ╪¿╪▒╪º█î ┘ç╪▒ ┘ü╪▒┘à╪¬ ╪»╪º╪»┘ç (JSON╪î XML╪î ╪¿╪º█î┘å╪▒█î) ╪»┘ê╪¿╪º╪▒┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪┤┘ê┘å╪» - **╪¿╪º╪▒ ┘å┌»┘ç╪»╪º╪▒█î**: ╪¬╪║█î█î╪▒╪º╪¬ ╪»╪▒ ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ╪»╪º╪»┘ç ┘å█î╪º╪▓ ╪¿┘ç ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î ╪»╪│╪¬█î ┌⌐╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪»╪º╪▒┘å╪» - **╪«╪╖╪º┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º**: ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪▒╪┤╪¬┘ç ╪º╪║┘ä╪¿ ╪»╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿┘ç ╪¼╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤┌⌐╪│╪¬ ┘à█î╪«┘ê╪▒┘å╪» ┘à╪º ┌å╪º╪▒┌å┘ê╪¿ ╪¼╪º┘à╪╣ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪¿╪▒╪º█î ╪▒┘ü╪╣ ╪º█î┘å ┌å╪º┘ä╪┤┘ç╪º ╪¬┘ê╪│╪╣┘ç ╪»╪º╪»█î┘à [111]. ### 6.1.2 ┘à╪╣┘à╪º╪▒█î ┘ê ╪╖╪▒╪º╪¡█î ┌å╪º╪▒┌å┘ê╪¿ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪º ╪º╪▓ ┘à╪╣┘à╪º╪▒█î ┌å┘å╪»┘ä╪º█î┘ç ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å╪»: cpp // ┘ä╪º█î┘ç 1: ┘à┘ê╪¬┘ê╪▒ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘ç╪│╪¬┘ç namespace serialization::core { template<typename T, typename Format> class reflection\_serializer { static\_assert(std::meta::is\_reflectable\_v<T>, "Type must be reflectable for serialization"); public: static typename Format::output\_type serialize(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); return serialize\_impl(obj, meta, Format{}); } static T deserialize(const typename Format::input\_type& data) { constexpr auto meta = std::meta::reflexpr(T); return deserialize\_impl(data, meta, Format{}); } private: template<std::meta::info Meta> static auto serialize\_impl(const T& obj, Meta meta, Format format) { typename Format::serialization\_context ctx; if constexpr (std::meta::is\_arithmetic\_v<meta>) { return format.serialize\_arithmetic(obj, ctx); } else if constexpr (std::meta::is\_class\_v<meta>) { return serialize\_class\_members(obj, meta, format, ctx); } else if constexpr (std::meta::is\_container\_v<meta>) { return serialize\_container(obj, meta, format, ctx); } } template<std::meta::info ClassMeta> static auto serialize\_class\_members(const T& obj, ClassMeta meta, Format format, auto& ctx) { constexpr auto members = std::meta::data\_members\_of(meta); format.begin\_object(ctx); std::meta::template\_for<members>([&](auto member\_meta) { constexpr auto name = std::meta::get\_name\_v<member\_meta>; constexpr auto member\_ptr = std::meta::get\_pointer\_v<member\_meta>; auto member\_value = obj.\*member\_ptr; auto serialized\_value = reflection\_serializer< std::remove\_cvref\_t<decltype(member\_value)>, Format >::serialize(member\_value); format.add\_member(ctx, name, serialized\_value); }); format.end\_object(ctx); return format.get\_result(ctx); } }; } // ┘ä╪º█î┘ç 2: ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪«╪º╪╡ ┘ü╪▒┘à╪¬ namespace serialization::formats { class json\_format { public: using output\_type = std::string; using input\_type = std::string\_view; struct serialization\_context { std::ostringstream stream; bool first\_member = true; }; template<typename T> void serialize\_arithmetic(const T& value, serialization\_context& ctx) { if constexpr (std::is\_same\_v<T, std::string>) { ctx.stream << '"' << escape\_json\_string(value) << '"'; } else { ctx.stream << value; } } void begin\_object(serialization\_context& ctx) { ctx.stream << '{'; ctx.first\_member = true; } void add\_member(serialization\_context& ctx, std::string\_view name, const std::string& value) { if (!ctx.first\_member) ctx.stream << ','; ctx.stream << '"' << name << '"' << ':' << value; ctx.first\_member = false; } void end\_object(serialization\_context& ctx) { ctx.stream << '}'; } std::string get\_result(serialization\_context& ctx) { return ctx.stream.str(); } }; class binary\_format { public: using output\_type = std::vector<uint8\_t>; using input\_type = std::span<const uint8\_t>; // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪º█î┘å╪▒█î }; class xml\_format { public: using output\_type = std::string; using input\_type = std::string\_view; // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î XML }; } // ┘ä╪º█î┘ç 3: █î┌⌐┘╛╪º╪▒┌å┌»█î ┘à╪¬╪º┌⌐┘ä╪º╪│ constexpr void serializable(std::meta::info target, auto... formats) { // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪▒╪º█î ┘ü╪▒┘à╪¬┘ç╪º█î ┘à╪┤╪«╪╡ ╪┤╪»┘ç (generate\_format\_methods(target, formats), ...); } template<typename Format> constexpr void generate\_format\_methods(std::meta::info target, Format format) { std::string class\_name = std::meta::get\_name\_v<target>; std::string format\_name = Format::name; std::string serialize\_method = "std::string to\_" + format\_name + "() const {\n" " return serialization::core::reflection\_serializer<" + class\_name + ", serialization::formats::" + format\_name + "\_format>::serialize(\*this);\n" "}\n"; std::string deserialize\_method = "static " + class\_name + " from\_" + format\_name + "(const std::string& data) {\n" " return serialization::core::reflection\_serializer<" + class\_name + ", serialization::formats::" + format\_name + "\_format>::deserialize(data);\n" "}\n"; std::meta::compiler.declare(target, serialize\_method); std::meta::compiler.declare(target, deserialize\_method); } ### 6.1.3 ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ╪º╪│╪¬┘ü╪º╪»┘ç ┌å╪º╪▒┌å┘ê╪¿ ┘ç┘à API ┘ç╪º█î ╪│╪╖╪¡ ┘╛╪º█î█î┘å ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘ç┘à ╪▒╪º╪¿╪╖┘ç╪º█î ╪│╪╖╪¡ ╪¿╪º┘ä╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»: cpp // ┘à╪½╪º┘ä 1: ╪º╪│╪¬┘ü╪º╪»┘ç ┘à╪│╪¬┘é█î┘à ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ struct Person { std::string name; int age; std::vector<std::string> hobbies; std::optional<std::string> email; }; // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪╡╪▒█î╪¡ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ std::string serialize\_person\_json(const Person& p) { return serialization::core::reflection\_serializer< Person, serialization::formats::json\_format >::serialize(p); } // ┘à╪½╪º┘ä 2: ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ class $serializable(json, xml, binary) Employee { int employee\_id; std::string name; std::string department; double salary; std::vector<std::string> skills; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - std::string to\_json() const // - std::string to\_xml() const // - std::vector<uint8\_t> to\_binary() const // - static Employee from\_json(const std::string&) // - static Employee from\_xml(const std::string&) // - static Employee from\_binary(std::span<const uint8\_t>) }; // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç Employee emp{1001, "Alice Johnson", "Engineering", 95000.0, {"C++", "Python"}}; // ┘à╪¬╪»┘ç╪º█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪º█î┘à┘å ┘å┘ê╪╣ ┘ê ┌⌐╪º╪▒╪º ┘ç╪│╪¬┘å╪» std::string json\_data = emp.to\_json(); std::string xml\_data = emp.to\_xml(); auto binary\_data = emp.to\_binary(); // ╪»█î╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä Employee restored = Employee::from\_json(json\_data); assert(emp.employee\_id == restored.employee\_id); ### 6.1.4 ╪º╪▒╪▓█î╪º╪¿█î ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪º ╪º╪▒╪▓█î╪º╪¿█î ╪¼╪º┘à╪╣ ╪╣┘à┘ä┌⌐╪▒╪» ┘à┘é╪º█î╪│┘ç ┌å╪º╪▒┌å┘ê╪¿ ╪«┘ê╪» ╪¿╪º ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à┘ê╪¼┘ê╪» ╪º┘å╪¼╪º┘à ╪»╪º╪»█î┘à [112]: **┘à┘é╪º█î╪│┘ç ╪╣┘à┘ä┌⌐╪▒╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î:** | ┌å╪º╪▒┌å┘ê╪¿ | ╪│╪▒█î╪º┘ä╪│╪º╪▓█î JSON | ╪»█î╪│╪▒█î╪º┘ä╪│╪º╪▓█î JSON | ╪¬╪ú╪½█î╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î | |ΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-| | nlohmann/json (╪»╪│╪¬█î) | 1,247ns ┬▒ 45ns | 2,134ns ┬▒ 78ns | +0KB | | Boost.Serialization | 2,891ns ┬▒ 112ns | 3,456ns ┬▒ 145ns | +245KB | | ┌å╪º╪▒┌å┘ê╪¿ ┘à╪º | 1,234ns ┬▒ 41ns | 2,098ns ┬▒ 72ns | +12KB | | **┘å╪│╪¿╪¬ ╪╣┘à┘ä┌⌐╪▒╪»** | **0.99x** | **0.98x** | **0.05x** | ┌å╪º╪▒┌å┘ê╪¿ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪º ╪╣┘à┘ä┌⌐╪▒╪»█î ╪▒┘é╪º╪¿╪¬█î ╪¿╪º ┌⌐╪» ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪┤╪»┘ç ╪»╪│╪¬█î ╪»╪º╪▒╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘å█î╪º╪▓ ╪¿┘ç ┘ç█î┌å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪»╪│╪¬█î ┘å╪»╪º╪▒╪». **┘à╪╣█î╪º╪▒┘ç╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç:** | ┘à╪╣█î╪º╪▒ | ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪»╪│╪¬█î | ┌å╪º╪▒┌å┘ê╪¿ ┘à╪º | ╪¿┘ç╪¿┘ê╪» | |ΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇö|ΓÇöΓÇöΓÇöΓÇö-| | ╪«╪╖┘ê╪╖ ┌⌐╪» | 342 LOC | 23 LOC | **┌⌐╪º┘ç╪┤ 93.3%** | | ╪▓┘à╪º┘å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î | 4.2 ╪│╪º╪╣╪¬ | 0.3 ╪│╪º╪╣╪¬ | **┌⌐╪º┘ç╪┤ 92.9%** | | ╪¿╪º┌»┘ç╪º█î ┘à╪╣╪▒┘ü█î ╪┤╪»┘ç | 7 ╪¿╪º┌» | 0 ╪¿╪º┌» | **┌⌐╪º┘ç╪┤ 100%** | | ╪¬┘ä╪º╪┤ ┘å┌»┘ç╪»╪º╪▒█î | ╪¿╪º┘ä╪º | ╪¡╪»╪º┘é┘ä | **╪¿┘ç╪¿┘ê╪» ┌⌐█î┘ü█î** | ## 6.2 ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ORM ### 6.2.1 ┌å╪º┘ä╪┤┘ç╪º█î ┘å┌»╪º╪┤╪¬ ╪ó╪¿╪¼┌⌐╪¬-╪▒╪º╪¿╪╖┘ç╪º█î ┘å┌»╪º╪┤╪¬ ╪ó╪¿╪¼┌⌐╪¬-╪▒╪º╪¿╪╖┘ç╪º█î (ORM) ╪»╪º┘à┘å┘ç ┘╛█î┌å█î╪»┘ç╪º█î ╪º╪│╪¬ ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º╪▒╪▓╪┤ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪» [113]. ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ╪│┘å╪¬█î ORM ╪¿╪º ┌å┘å╪»█î┘å ┌å╪º┘ä╪┤ ┘à┘ê╪º╪¼┘ç ┘ç╪│╪¬┘å╪»: - **┘ç┘à┌»╪º┘à╪│╪º╪▓█î ╪╖╪▒╪¡┘ê╪º╪▒┘ç**: ┘å┌»┘ç╪»╪º╪┤╪¬┘å ╪╖╪▒╪¡┘ê╪º╪▒┘ç┘ç╪º█î ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ┘ç┘à┌»╪º┘à ╪¿╪º ╪¬╪╣╪º╪▒█î┘ü ╪ó╪¿╪¼┌⌐╪¬ - **╪º█î┘à┘å█î ┘å┘ê╪╣**: ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪╣┘à┘ä█î╪º╪¬ ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç - **╪╣┘à┘ä┌⌐╪▒╪»**: ┌⌐┘à█î┘å┘ç ┌⌐╪▒╪»┘å ╪│╪▒╪¿╪º╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪»╪▒ ╪¡█î┘å ╪¡┘ü╪╕ ╪º┘å╪╣╪╖╪º┘ü┘╛╪░█î╪▒█î - **╪¬┘ê┘ä█î╪» ┌⌐╪»**: ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪╣┘à┘ä█î╪º╪¬ CRUD ┘ê ╪│╪º╪▓┘å╪»┘ç┘ç╪º█î ┌⌐┘ê╪ª╪▒█î ### 6.2.2 ╪╖╪▒╪º╪¡█î ORM ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ORM ┘à╪º ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪╖╪▒╪¡┘ê╪º╪▒┘ç ┘ê ╪│╪º╪«╪¬ ┌⌐┘ê╪ª╪▒█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å╪»: cpp // ╪▓█î╪▒╪│╪º╪«╪¬ ┘ç╪│╪¬┘ç ORM ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ namespace orm::core { template<typename Entity> class entity\_mapper { static\_assert(std::meta::has\_metaclass<entity>(Entity), "Type must use entity metaclass"); public: using primary\_key\_type = typename detect\_primary\_key<Entity>::type; static std::string get\_table\_name() { constexpr auto meta = std::meta::reflexpr(Entity); return std::meta::get\_attribute\_v<table\_name>(meta); } static std::string generate\_create\_table\_sql() { constexpr auto meta = std::meta::reflexpr(Entity); constexpr auto members = std::meta::data\_members\_of(meta); std::ostringstream sql; sql << "CREATE TABLE " << get\_table\_name() << " (\n"; bool first = true; std::meta::template\_for<members>([&](auto member\_meta) { if (!first) sql << ",\n"; first = false; constexpr auto name = std::meta::get\_name\_v<member\_meta>; constexpr auto type = std::meta::get\_type\_t<member\_meta>; sql << " " << name << " " << map\_cpp\_type\_to\_sql<type>(); if constexpr (std::meta::has\_attribute<primary\_key>(member\_meta)) { sql << " PRIMARY KEY"; } if constexpr (std::meta::has\_attribute<not\_null>(member\_meta)) { sql << " NOT NULL"; } if constexpr (std::meta::has\_attribute<unique>(member\_meta)) { sql << " UNIQUE"; } }); sql << "\n);"; return sql.str(); } static Entity from\_result\_set(const database::result\_row& row) { Entity entity; constexpr auto meta = std::meta::reflexpr(Entity); constexpr auto members = std::meta::data\_members\_of(meta); size\_t column\_index = 0; std::meta::template\_for<members>([&](auto member\_meta) { constexpr auto member\_ptr = std::meta::get\_pointer\_v<member\_meta>; constexpr auto member\_type = std::meta::get\_type\_t<member\_meta>; entity.\*member\_ptr = row.get<member\_type>(column\_index++); }); return entity; } static std::vector<std::string> get\_column\_names() { constexpr auto meta = std::meta::reflexpr(Entity); constexpr auto members = std::meta::data\_members\_of(meta); std::vector<std::string> columns; std::meta::template\_for<members>([&](auto member\_meta) { constexpr auto name = std::meta::get\_name\_v<member\_meta>; columns.emplace\_back(name); }); return columns; } }; // ╪│╪º╪▓┘å╪»┘ç ┌⌐┘ê╪ª╪▒█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename Entity> class query\_builder { std::ostringstream query\_; std::vector<database::parameter> parameters\_; public: query\_builder() { query\_ << "SELECT \* FROM " << entity\_mapper<Entity>::get\_table\_name(); } template<auto MemberPtr> query\_builder& where(const auto& value) { constexpr auto member\_meta = std::meta::reflexpr(MemberPtr); constexpr auto column\_name = std::meta::get\_name\_v<member\_meta>; if (parameters\_.empty()) { query\_ << " WHERE "; } else { query\_ << " AND "; } query\_ << column\_name << " = ?"; parameters\_.emplace\_back(value); return \*this; } template<auto MemberPtr> query\_builder& order\_by(sort\_direction direction = ascending) { constexpr auto member\_meta = std::meta::reflexpr(MemberPtr); constexpr auto column\_name = std::meta::get\_name\_v<member\_meta>; query\_ << " ORDER BY " << column\_name; if (direction == descending) { query\_ << " DESC"; } return \*this; } std::vector<Entity> execute(database::connection& conn) { auto result = conn.execute(query\_.str(), parameters\_); std::vector<Entity> entities; for (const auto& row : result) { entities.push\_back(entity\_mapper<Entity>::from\_result\_set(row)); } return entities; } }; } // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» entity constexpr void entity(std::meta::info target, std::string\_view table\_name = "", bool generate\_crud = true) { // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º┘ä╪▓╪º┘à╪º╪¬ entity validate\_entity\_constraints(target); // ╪¬┘ê┘ä█î╪» ┘ü╪▒╪º╪»╪º╪»┘ç ┘å┌»╪º╪┤╪¬ ╪¼╪»┘ê┘ä generate\_table\_metadata(target, table\_name); if (generate\_crud) { generate\_crud\_methods(target); } // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î ╪│╪º╪▓┘å╪»┘ç ┌⌐┘ê╪ª╪▒█î generate\_query\_methods(target); } constexpr void generate\_crud\_methods(std::meta::info target) { std::string class\_name = std::meta::get\_name\_v<target>; // ╪¬┘ê┘ä█î╪» ┘à╪¬╪» save std::string save\_method = R"( void save(orm::database::connection& conn) { auto mapper = orm::core::entity\_mapper<)" + class\_name + R"(>{}; if ()" + get\_primary\_key\_member\_name(target) + R"( == 0) { // ╪»╪▒╪¼ ╪▒┌⌐┘ê╪▒╪» ╪¼╪»█î╪» insert(conn); } else { // ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î ╪▒┌⌐┘ê╪▒╪» ┘à┘ê╪¼┘ê╪» update(conn); } } void insert(orm::database::connection& conn) { // ╪»╪│╪¬┘ê╪▒ INSERT ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪º╪▓╪¬╪º╪¿ } void update(orm::database::connection& conn) { // ╪»╪│╪¬┘ê╪▒ UPDATE ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪º╪▓╪¬╪º╪¿ } void remove(orm::database::connection& conn) { // ╪»╪│╪¬┘ê╪▒ DELETE ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪º╪▓╪¬╪º╪¿ } )"; std::meta::compiler.declare(target, save\_method); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î finder ╪º╪│╪¬╪º╪¬█î┌⌐ std::string finder\_methods = R"( static std::optional<)" + class\_name + R"(> find( orm::database::connection& conn, const auto& primary\_key) { // SELECT ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ┌⌐┘ä█î╪» ╪º╪╡┘ä█î } static std::vector<)" + class\_name + R"(> find\_all( orm::database::connection& conn) { return orm::core::query\_builder<)" + class\_name + R"(>{} .execute(conn); } template<auto MemberPtr> static std::vector<)" + class\_name + R"(> find\_by( orm::database::connection& conn, const auto& value) { return orm::core::query\_builder<)" + class\_name + R"(>{} .where<MemberPtr>(value) .execute(conn); } )"; std::meta::compiler.declare(target, finder\_methods); } ### 6.2.3 ┘à╪½╪º┘ä┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┘ê ╪º█î┘à┘å█î ┘å┘ê╪╣ ORM ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»: cpp // ╪¬╪╣╪▒█î┘ü entity ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º class $entity("users") User { $primary\_key int id; $unique std::string email; $not\_null std::string name; std::optional<std::string> bio; std::chrono::system\_clock::time\_point created\_at; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - SQL ╪º█î╪¼╪º╪» ╪¼╪»┘ê┘ä // - ╪╣┘à┘ä█î╪º╪¬ CRUD (save, insert, update, remove) // - ╪│╪º╪▓┘å╪»┘ç┘ç╪º█î ┌⌐┘ê╪ª╪▒█î ╪º█î┘à┘å ┘å┘ê╪╣ // - ┘å┌»╪º╪┤╪¬ ┘à╪¼┘à┘ê╪╣┘ç ┘å╪¬╪º█î╪¼ }; class $entity("posts") Post { $primary\_key int id; $foreign\_key("users", "id") int user\_id; $not\_null std::string title; std::string content; std::chrono::system\_clock::time\_point published\_at; }; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä void demonstrate\_orm\_usage() { orm::database::connection conn("postgresql://localhost/mydb"); // ╪º█î╪¼╪º╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪¼╪»╪º┘ê┘ä conn.execute(User::get\_create\_table\_sql()); conn.execute(Post::get\_create\_table\_sql()); // ╪╣┘à┘ä█î╪º╪¬ entity ╪º█î┘à┘å ┘å┘ê╪╣ User user{0, "alice@example.com", "Alice Johnson", "Software Engineer"}; user.save(conn); // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿█î┘å INSERT ┘ê UPDATE ╪¬╪┤╪«█î╪╡ ┘à█î╪»┘ç╪» // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪¿╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä auto users\_named\_alice = User::find\_by<&User::name>(conn, "Alice Johnson"); auto user\_by\_email = User::find\_by<&User::email>(conn, "alice@example.com"); // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪¿╪º ╪▒╪º╪¿╪╖ ╪▒┘ê╪º┘å auto recent\_posts = orm::core::query\_builder<Post>{} .where<&Post::user\_id>(user.id) .order\_by<&Post::published\_at>(orm::descending) .execute(conn); // ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪«╪╖╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä // auto invalid = User::find\_by<&Post::title>(conn, "test"); // ╪«╪╖╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä! // user.nonexistent\_field = "value"; // ╪«╪╖╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä! } ### 6.2.4 ┘à╪▓╪º█î╪º█î ╪╣┘à┘ä┌⌐╪▒╪» ┘ê ┘à┘ç╪º╪¼╪▒╪¬ **┘à┘é╪º█î╪│┘ç ╪╣┘à┘ä┌⌐╪▒╪» ORM:** | ┌å╪º╪▒┌å┘ê╪¿ | ╪º╪¼╪▒╪º█î ┌⌐┘ê╪ª╪▒█î | ┘å┌»╪º╪┤╪¬ ╪ó╪¿╪¼┌⌐╪¬ | ╪º╪│╪¬┘ü╪º╪»┘ç ╪¡╪º┘ü╪╕┘ç | ╪º█î┘à┘å█î ┘å┘ê╪╣ | |ΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇö-|ΓÇöΓÇöΓÇöΓÇöΓÇô|ΓÇöΓÇöΓÇöΓÇö-| | SQL ╪│┘å╪¬█î | 1.0x (┘à╪¿┘å╪º) | ╪»╪│╪¬█î | ┘╛╪º█î█î┘å | ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | Hibernate OGM (Java) | 1.8x | ╪«┘ê╪»┌⌐╪º╪▒ | ╪¿╪º┘ä╪º | ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | Django ORM (Python) | 2.3x | ╪«┘ê╪»┌⌐╪º╪▒ | ┘à╪¬┘ê╪│╪╖ | ╪▓┘à╪º┘å ╪º╪¼╪▒╪º | | ORM ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪º | 1.1x | ╪«┘ê╪»┌⌐╪º╪▒ | ┘╛╪º█î█î┘å | **╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä** | **┘à┘ç╪º╪¼╪▒╪¬ ┘ê ╪¬┌⌐╪º┘à┘ä ╪╖╪▒╪¡┘ê╪º╪▒┘ç:** cpp // ┘å╪│╪«┘ç 1 entity ┌⌐╪º╪▒╪¿╪▒ class $entity("users") $version(1) User { $primary\_key int id; std::string name; std::string email; }; // ┘å╪│╪«┘ç 2 ╪¿╪º ┘ü█î┘ä╪»┘ç╪º█î ╪º╪╢╪º┘ü█î class $entity("users") $version(2) User { $primary\_key int id; std::string first\_name; // ╪¬┘é╪│█î┘à ╪º╪▓ name std::string last\_name; // ╪¬┘é╪│█î┘à ╪º╪▓ name std::string email; std::optional<std::string> phone; // ┘ü█î┘ä╪» ╪¼╪»█î╪» // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à┘ç╪º╪¼╪▒╪¬ static void migrate\_from\_v1(database::connection& conn) { // ┘à┘å╪╖┘é ┘à┘ç╪º╪¼╪▒╪¬ ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¬┘ü╪º┘ê╪¬ ╪╖╪▒╪¡┘ê╪º╪▒┘ç } }; ## 6.3 ┌å╪º╪▒┌å┘ê╪¿ GUI ╪¿╪º ╪º╪¬╪╡╪º┘ä ╪«┘ê╪»┌⌐╪º╪▒ ### 6.3.1 ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º╪╣┘ä╪º┘å█î UI ╪¬┘ê╪│╪╣┘ç GUI ┘à╪»╪▒┘å ╪¿┘ç ╪╖┘ê╪▒ ┘ü╪▓╪º█î┘å╪»┘ç╪º█î ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪º╪╣┘ä╪º┘å█î ╪▒╪º ╪¬╪▒╪¼█î╪¡ ┘à█î╪»┘ç╪» ┌⌐┘ç ╪»╪▒ ╪ó┘å ╪│╪º╪«╪¬╪º╪▒ ┘ê ╪▒┘ü╪¬╪º╪▒ UI ┘à╪┤╪«╪╡ ┘à█î╪┤┘ê╪» ╪¿┘ç ╪¼╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º┘à╪▒█î [114]. ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î GUI ╪º╪╣┘ä╪º┘å█î ┘é╪»╪▒╪¬┘à┘å╪» ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪»: cpp // ┌å╪º╪▒┌å┘ê╪¿ UI ╪º╪╣┘ä╪º┘å█î ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º namespace gui::declarative { // ╪│█î╪│╪¬┘à ┘ê█î╪¼╪¬ ┘╛╪º█î┘ç ╪¿╪º ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> concept Widget = requires { typename T::properties\_type; std::meta::is\_reflectable\_v<T>; }; // ╪│█î╪│╪¬┘à ╪º╪¬╪╡╪º┘ä ╪«╪╡┘ê╪╡█î╪¬ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename SourceType, typename TargetWidget> class property\_binding { static\_assert(Widget<TargetWidget>); SourceType\* source\_; TargetWidget\* target\_; std::vector<std::function<void()>> update\_callbacks\_; public: template<auto SourceMember, auto TargetProperty> void bind() { constexpr auto source\_meta = std::meta::reflexpr(SourceMember); constexpr auto target\_meta = std::meta::reflexpr(TargetProperty); static\_assert(std::is\_same\_v< std::meta::get\_type\_t<source\_meta>, std::meta::get\_type\_t<target\_meta> >, "Bound properties must have compatible types"); // ╪º█î╪¼╪º╪» ╪º╪¬╪╡╪º┘ä ╪»┘ê╪╖╪▒┘ü┘ç auto update\_target = [this]() { target\_->\*TargetProperty = source\_->\*SourceMember; target\_->update(); }; auto update\_source = [this]() { source\_->\*SourceMember = target\_->\*TargetProperty; // ┘ü╪╣╪º┘ä╪│╪º╪▓█î ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î┘ç╪º█î ╪ó╪¿╪¼┌⌐╪¬ ┘à╪¿╪»╪ú if constexpr (std::meta::has\_method<notify\_property\_changed>( std::meta::reflexpr(SourceType))) { constexpr auto property\_name = std::meta::get\_name\_v<source\_meta>; source\_->notify\_property\_changed(property\_name); } }; update\_callbacks\_.push\_back(update\_target); target\_->on\_property\_changed(TargetProperty, update\_source); // ┘ç┘à┌»╪º┘à╪│╪º╪▓█î ╪º┘ê┘ä█î┘ç update\_target(); } }; } // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪«╪╡┘ê╪╡█î╪¬ UI constexpr void ui\_model(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪│█î╪│╪¬┘à ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î ╪¬╪║█î█î╪▒ ╪«╪╡┘ê╪╡█î╪¬ generate\_property\_notifications(target); // ╪¬┘ê┘ä█î╪» ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪«╪╡┘ê╪╡█î╪¬ generate\_property\_validation(target); // ╪¬┘ê┘ä█î╪» ┌⌐┘à┌⌐┌⌐┘å┘å╪»┘ç┘ç╪º█î ╪º╪¬╪╡╪º┘ä UI generate\_binding\_methods(target); } constexpr void generate\_property\_notifications(std::meta::info target) { std::string notification\_system = R"( private: std::unordered\_map<std::string, std::vector<std::function<void()>>> property\_observers\_; public: void add\_property\_observer(const std::string& property\_name, std::function<void()> observer) { property\_observers\_[property\_name].push\_back(std::move(observer)); } void notify\_property\_changed(const std::string& property\_name) { auto it = property\_observers\_.find(property\_name); if (it != property\_observers\_.end()) { for (const auto& observer : it->second) { observer(); } } } )"; std::meta::compiler.declare(target, notification\_system); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î setter ╪¿╪º ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î ╪¿╪▒╪º█î ╪¬┘à╪º┘à ╪º╪╣╪╢╪º constexpr auto members = std::meta::data\_members\_of(target); std::meta::template\_for<members>([&](auto member\_meta) { generate\_notifying\_setter(target, member\_meta); }); } // ┘à╪½╪º┘ä ╪¿╪▒┘å╪º┘à┘ç GUI ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪º╪¬╪╡╪º┘ä ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ class $ui\_model PersonViewModel { std::string name; int age; std::string email; bool is\_verified; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î┘ç╪º█î ╪¬╪║█î█î╪▒ ╪«╪╡┘ê╪╡█î╪¬ // - ┘à╪¬╪»┘ç╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î // - ┌⌐┘à┌⌐┌⌐┘å┘å╪»┘ç┘ç╪º█î ╪º╪¬╪╡╪º┘ä UI // - ┘à╪¬╪»┘ç╪º█î getter/setter ╪¿╪º ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î }; class PersonEditDialog : public gui::Dialog { gui::TextEdit name\_edit\_; gui::SpinBox age\_spinbox\_; gui::LineEdit email\_edit\_; gui::CheckBox verified\_checkbox\_; PersonViewModel\* model\_; gui::declarative::property\_binding<PersonViewModel, PersonEditDialog> binding\_; public: PersonEditDialog(PersonViewModel\* model) : model\_(model), binding\_(model, this) { // ╪º╪¬╪╡╪º┘ä ╪«┘ê╪»┌⌐╪º╪▒ ╪«╪╡┘ê╪╡█î╪¬ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ binding\_.bind<&PersonViewModel::name, &PersonEditDialog::name\_edit\_>(); binding\_.bind<&PersonViewModel::age, &PersonEditDialog::age\_spinbox\_>(); binding\_.bind<&PersonViewModel::email, &PersonEditDialog::email\_edit\_>(); binding\_.bind<&PersonViewModel::is\_verified, &PersonEditDialog::verified\_checkbox\_>(); // ╪¬┘à╪º┘à ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î┘ç╪º█î UI ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪º ┘à╪»┘ä ┘ç┘à┌»╪º┘à╪│╪º╪▓█î ┘à█î╪┤┘ê┘å╪» } }; ### 6.3.2 ╪¬┘ê┘ä█î╪» ┘ü╪▒┘à ┘ê ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌å╪º╪▒┌å┘ê╪¿ ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ┘ü╪▒┘à┘ç╪º ╪▒╪º ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘à╪»┘ä ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: cpp // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘ü╪▒┘à ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º constexpr void form\_generator(std::meta::info target, gui::layout\_type layout = gui::vertical) { // ╪¬╪¡┘ä█î┘ä ╪│╪º╪«╪¬╪º╪▒ ┘à╪»┘ä constexpr auto members = std::meta::data\_members\_of(target); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪» ╪º█î╪¼╪º╪» ┘ü╪▒┘à std::string form\_method = R"( std::unique\_ptr<gui::Form> create\_form() const { auto form = std::make\_unique<gui::Form>(); form->set\_layout()" + std::to\_string(static\_cast<int>(layout)) + R"(); )"; std::meta::template\_for<members>([&](auto member\_meta) { constexpr auto member\_type = std::meta::get\_type\_t<member\_meta>; constexpr auto member\_name = std::meta::get\_name\_v<member\_meta>; if constexpr (std::is\_same\_v<member\_type, std::string>) { form\_method += "form->add\_text\_field(\"" + std::string(member\_name) + "\");\n"; } else if constexpr (std::is\_same\_v<member\_type, int>) { form\_method += "form->add\_number\_field(\"" + std::string(member\_name) + "\");\n"; } else if constexpr (std::is\_same\_v<member\_type, bool>) { form\_method += "form->add\_checkbox(\"" + std::string(member\_name) + "\");\n"; } else if constexpr (std::is\_same\_v<member\_type, std::chrono::system\_clock::time\_point>) { form\_method += "form->add\_date\_field(\"" + std::string(member\_name) + "\");\n"; } }); form\_method += R"( return form; } )"; std::meta::compiler.declare(target, form\_method); } // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç class $ui\_model $form\_generator(gui::grid\_layout) EmployeeRecord { $required std::string employee\_id; $required std::string first\_name; $required std::string last\_name; $email\_validation std::string email; $range(18, 65) int age; std::string department; $currency double salary; std::chrono::system\_clock::time\_point hire\_date; $multiline std::string notes; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - ┘ü╪▒┘à ╪¿╪º ┘ê█î╪¼╪¬┘ç╪º█î ┘à┘å╪º╪│╪¿ ╪¿╪▒╪º█î ┘ç╪▒ ┘ü█î┘ä╪» // - ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¿╪▒ ╪º╪│╪º╪│ ┘ê█î┌ÿ┌»█î┘ç╪º // - ╪º╪¬╪╡╪º┘ä ╪»╪º╪»┘ç ╪¿█î┘å ┘ü╪▒┘à ┘ê ┘à╪»┘ä // - ┘å┘à╪º█î╪┤ ┘ê ┘à╪»█î╪▒█î╪¬ ╪«╪╖╪º }; ## 6.4 ╪¬┘ê┘ä█î╪» ┌å╪º╪▒┌å┘ê╪¿ ╪ó╪▓┘à┘ê┘å ### 6.4.1 ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à┘ê╪▒╪» ╪ó╪▓┘à┘ê┘å ╪ó╪▓┘à╪º█î╪┤ ╪»╪º┘à┘å┘ç ╪»█î┌»╪▒█î ╪º╪│╪¬ ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪▒╪▓╪┤ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿╪º ╪º┘à┌⌐╪º┘å ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪ó╪▓┘à┘ê┘å ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [115]: cpp // ┌å╪º╪▒┌å┘ê╪¿ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪ó╪▓┘à┘ê┘å namespace testing::reflection { template<typename TestClass> class test\_suite\_generator { static\_assert(std::meta::has\_metaclass<test\_suite>(TestClass)); public: static void generate\_and\_run\_tests() { constexpr auto meta = std::meta::reflexpr(TestClass); constexpr auto methods = std::meta::member\_functions\_of(meta); TestClass test\_instance; // ┘ü╪º╪▓ ╪▒╪º┘ç╪º┘å╪»╪º╪▓█î if constexpr (std::meta::has\_method<setup>(meta)) { test\_instance.setup(); } // ╪º╪¼╪▒╪º█î ╪¬┘à╪º┘à ┘à╪¬╪»┘ç╪º█î ╪ó╪▓┘à┘ê┘å std::meta::template\_for<methods>([&](auto method\_meta) { constexpr auto method\_name = std::meta::get\_name\_v<method\_meta>; if constexpr (method\_name.starts\_with("test\_")) { execute\_test\_method(test\_instance, method\_meta); } }); // ┘ü╪º╪▓ ╪¬┘à█î╪▓┌⌐╪º╪▒█î if constexpr (std::meta::has\_method<teardown>(meta)) { test\_instance.teardown(); } } private: template<auto MethodMeta> static void execute\_test\_method(TestClass& instance, MethodMeta method) { constexpr auto method\_name = std::meta::get\_name\_v<method>; constexpr auto method\_ptr = std::meta::get\_pointer\_v<method>; try { std::cout << "Running test: " << method\_name << "... "; // ╪º╪¼╪▒╪º█î ┘à╪¬╪» ╪ó╪▓┘à┘ê┘å (instance.\*method\_ptr)(); std::cout << "PASSED\n"; } catch (const testing::assertion\_failed& e) { std::cout << "FAILED: " << e.what() << "\n"; } catch (const std::exception& e) { std::cout << "ERROR: " << e.what() << "\n"; } } }; } // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┘à╪¼┘à┘ê╪╣┘ç ╪ó╪▓┘à┘ê┘å constexpr void test\_suite(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪▓█î╪▒╪│╪º╪«╪¬ ╪º╪¼╪▒╪º█î ╪ó╪▓┘à┘ê┘å generate\_test\_runner(target); // ╪¬┘ê┘ä█î╪» ┌⌐┘à┌⌐┌⌐┘å┘å╪»┘ç┘ç╪º█î assertion ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘å┘ê╪º╪╣ ╪╣╪╢┘ê generate\_assertion\_helpers(target); // ╪¬┘ê┘ä█î╪» ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪ó╪¿╪¼┌⌐╪¬ mock generate\_mock\_support(target); } // ┘à╪½╪º┘ä ┌⌐┘ä╪º╪│ ╪ó╪▓┘à┘ê┘å ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪ó╪▓┘à╪º█î╪┤ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ class $test\_suite CalculatorTests { Calculator calc\_; public: void setup() { calc\_.reset(); } void teardown() { // ╪¬┘à█î╪▓┌⌐╪º╪▒█î ╪»╪▒ ╪╡┘ê╪▒╪¬ ┘å█î╪º╪▓ } void test\_addition() { auto result = calc\_.add(2, 3); assert\_equals(5, result); } void test\_division\_by\_zero() { assert\_throws<std::domain\_error>([&]() { calc\_.divide(10, 0); }); } void test\_complex\_calculation() { calc\_.add(10, 5); calc\_.multiply(2); calc\_.subtract(5); assert\_equals(25, calc\_.get\_result()); } // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - ┌⌐╪┤┘ü ┘ê ╪º╪¼╪▒╪º█î ╪ó╪▓┘à┘ê┘å // - ┘à╪»█î╪▒█î╪¬ setup/teardown // - ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º ┘ê ╪ó┘à╪º╪▒ // - █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪º╪¼╪▒╪º┌⌐┘å┘å╪»┘ç┘ç╪º█î ╪ó╪▓┘à┘ê┘å }; ### 6.4.2 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪ó╪▓┘à╪º█î╪┤ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪«╪╡┘ê╪╡█î╪¬ ┌å╪º╪▒┌å┘ê╪¿ ╪º╪▓ ╪ó╪▓┘à╪º█î╪┤ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪«╪╡┘ê╪╡█î╪¬ ╪¿╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à┘ê╪▒╪» ╪ó╪▓┘à┘ê┘å ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à█î┌⌐┘å╪»: cpp // ╪ó╪▓┘à╪º█î╪┤ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪«╪╡┘ê╪╡█î╪¬ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> class property\_test\_generator { public: template<auto Property> static void test\_property(size\_t num\_iterations = 1000) { constexpr auto prop\_meta = std::meta::reflexpr(Property); constexpr auto param\_types = std::meta::get\_parameter\_types\_t<prop\_meta>; for (size\_t i = 0; i < num\_iterations; ++i) { auto test\_inputs = generate\_random\_inputs<param\_types>(); try { bool result = std::apply(Property, test\_inputs); if (!result) { report\_property\_violation(Property, test\_inputs); } } catch (const std::exception& e) { report\_property\_exception(Property, test\_inputs, e); } } } private: template<typename... Types> static std::tuple<Types...> generate\_random\_inputs() { return std::make\_tuple(generate\_random\_value<Types>()...); } template<typename Type> static Type generate\_random\_value() { if constexpr (std::is\_integral\_v<Type>) { return random\_distribution<Type>()(); } else if constexpr (std::is\_floating\_point\_v<Type>) { return random\_distribution<Type>()(); } else if constexpr (std::is\_same\_v<Type, std::string>) { return generate\_random\_string(); } // ╪¬┘ê┘ä█î╪»┌⌐┘å┘å╪»┘ç┘ç╪º█î ╪«╪º╪╡ ┘å┘ê╪╣ ╪¿█î╪┤╪¬╪▒ ╪º╪╢╪º┘ü┘ç ┌⌐┘å█î╪» } }; // ┘à╪½╪º┘ä ╪ó╪▓┘à┘ê┘å┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪«╪╡┘ê╪╡█î╪¬ class MathProperties { public: static bool addition\_commutative(int a, int b) { return (a + b) == (b + a); } static bool multiplication\_associative(int a, int b, int c) { return (a \* (b \* c)) == ((a \* b) \* c); } static bool sort\_idempotent(std::vector<int> vec) { auto sorted1 = vec; std::sort(sorted1.begin(), sorted1.end()); auto sorted2 = sorted1; std::sort(sorted2.begin(), sorted2.end()); return sorted1 == sorted2; } }; // ╪ó╪▓┘à╪º█î╪┤ ╪«┘ê╪»┌⌐╪º╪▒ ╪«╪╡┘ê╪╡█î╪¬ void run\_property\_tests() { property\_test\_generator<MathProperties>::test\_property< &MathProperties::addition\_commutative>(10000); property\_test\_generator<MathProperties>::test\_property< &MathProperties::multiplication\_associative>(10000); property\_test\_generator<MathProperties>::test\_property< &MathProperties::sort\_idempotent>(1000); } ## 6.5 ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ### 6.5.1 ╪º┘ä┌»┘ê█î Observer ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘╛█î┌å█î╪»┘ç ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» [116]: cpp // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º┘ä┌»┘ê█î observer constexpr void observable(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪▓█î╪▒╪│╪º╪«╪¬ observer std::string observer\_infrastructure = R"( private: mutable std::unordered\_map<std::string, std::vector<std::function<void(const std::any&)>>> observers\_; public: template<typename T> void add\_observer(const std::string& property\_name, std::function<void(const T&)> callback) { observers\_[property\_name].emplace\_back([callback](const std::any& value) { callback(std::any\_cast<const T&>(value)); }); } void remove\_all\_observers(const std::string& property\_name = "") { if (property\_name.empty()) { observers\_.clear(); } else { observers\_.erase(property\_name); } } protected: template<typename T> void notify\_observers(const std::string& property\_name, const T& value) { auto it = observers\_.find(property\_name); if (it != observers\_.end()) { for (const auto& observer : it->second) { observer(std::make\_any<T>(value)); } } } )"; std::meta::compiler.declare(target, observer\_infrastructure); // ╪¬┘ê┘ä█î╪» setter ┘ç╪º█î ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å ╪¿╪▒╪º█î ╪¬┘à╪º┘à ╪º╪╣╪╢╪º constexpr auto members = std::meta::data\_members\_of(target); std::meta::template\_for<members>([&](auto member\_meta) { generate\_notifying\_setter(target, member\_meta); }); } // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç class $observable $serializable StockPrice { std::string symbol; double price; double volume; std::chrono::system\_clock::time\_point timestamp; // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // - ┘à╪¬╪»┘ç╪º█î ╪½╪¿╪¬/╪¡╪░┘ü observer // - ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î┘ç╪º█î ╪«┘ê╪»┌⌐╪º╪▒ ╪»╪▒ ╪¬╪║█î█î╪▒╪º╪¬ ╪«╪╡┘ê╪╡█î╪¬ // - callback ┘ç╪º█î observer ╪º█î┘à┘å ┘å┘ê╪╣ }; // ╪º╪│╪¬┘ü╪º╪»┘ç observer void demonstrate\_observer\_pattern() { StockPrice stock; // ╪½╪¿╪¬ observer ┘ç╪º ╪¿╪▒╪º█î ╪«╪╡┘ê╪╡█î╪¬┘ç╪º█î ╪«╪º╪╡ stock.add\_observer<double>("price", [](const double& new\_price) { std::cout << "Price changed to: $" << new\_price << std::endl; }); stock.add\_observer<double>("volume", [](const double& new\_volume) { std::cout << "Volume changed to: " << new\_volume << std::endl; }); // ╪¬╪║█î█î╪▒╪º╪¬ ╪«╪╡┘ê╪╡█î╪¬ ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪╖┘ä╪º╪╣╪▒╪│╪º┘å█î┘ç╪º ╪▒╪º ┘ü╪╣╪º┘ä ┘à█î┌⌐┘å┘å╪» stock.set\_price(150.75); // observer ┘é█î┘à╪¬ ╪▒╪º ┘ü╪╣╪º┘ä ┘à█î┌⌐┘å╪» stock.set\_volume(1000000); // observer ╪¡╪¼┘à ╪▒╪º ┘ü╪╣╪º┘ä ┘à█î┌⌐┘å╪» } ### 6.5.2 ╪º╪¬┘ê┘à╪º╪│█î┘ê┘å ╪º┘ä┌»┘ê█î Visitor ╪º┘ä┌»┘ê┘ç╪º█î ╪│┘ä╪│┘ä┘ç┘à╪▒╪º╪¬╪¿█î ┘╛█î┌å█î╪»┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪«┘ê╪»┌⌐╪º╪▒ ╪┤┘ê┘å╪»: cpp // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê█î visitor constexpr void visitable(std::meta::info target) { std::meta::compiler.require(std::meta::is\_polymorphic\_v<target>, "visitable requires polymorphic type"); // ╪¬┘ê┘ä█î╪» ╪▒╪º╪¿╪╖ visitor generate\_visitor\_interface(target); // ╪¬┘ê┘ä█î╪» ┘à╪¬╪» accept std::string accept\_method = R"( template<typename Visitor> auto accept(Visitor&& visitor) const { return visitor.visit(\*this); } template<typename Visitor> auto accept(Visitor&& visitor) { return visitor.visit(\*this); } )"; std::meta::compiler.declare(target, accept\_method); } // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪▒╪º╪¿╪╖ visitor constexpr void generate\_visitor\_interface(std::meta::info base\_type) { auto derived\_types = std::meta::get\_derived\_types\_t<base\_type>; std::string visitor\_interface = "template<typename ReturnType = void>\n"; visitor\_interface += "class " + std::meta::get\_name\_v<base\_type> + "Visitor {\n"; visitor\_interface += "public:\n"; // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î visit ╪¿╪▒╪º█î ┘ç╪▒ ┘å┘ê╪╣ ┘à╪┤╪¬┘é ╪┤╪»┘ç std::meta::template\_for<derived\_types>([&](auto derived\_meta) { auto type\_name = std::meta::get\_name\_v<derived\_meta>; visitor\_interface += " virtual ReturnType visit(const " + type\_name + "&) = 0;\n"; visitor\_interface += " virtual ReturnType visit(" + type\_name + "&) = 0;\n"; }); visitor\_interface += "};\n"; // ╪¬╪▓╪▒█î┘é ╪▒╪º╪¿╪╖ visitor ╪¿┘ç ┘ü╪╢╪º█î ┘å╪º┘à ╪│╪▒╪º╪│╪▒█î std::meta::compiler.declare\_global(visitor\_interface); } // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç class $visitable Shape { public: virtual ~Shape() = default; virtual double area() const = 0; }; class Circle : public Shape { double radius\_; public: Circle(double r) : radius\_(r) {} double area() const override { return M\_PI \* radius\_ \* radius\_; } double get\_radius() const { return radius\_; } }; class Rectangle : public Shape { double width\_, height\_; public: Rectangle(double w, double h) : width\_(w), height\_(h) {} double area() const override { return width\_ \* height\_; } double get\_width() const { return width\_; } double get\_height() const { return height\_; } }; // ╪▒╪º╪¿╪╖ visitor ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪«┘ê╪»┌⌐╪º╪▒: // template<typename ReturnType = void> // class ShapeVisitor { // public: // virtual ReturnType visit(const Circle&) = 0; // virtual ReturnType visit(Circle&) = 0; // virtual ReturnType visit(const Rectangle&) = 0; // virtual ReturnType visit(Rectangle&) = 0; // }; // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î visitor ┘à╪┤╪«╪╡ class AreaCalculatorVisitor : public ShapeVisitor<double> { public: double visit(const Circle& circle) override { return circle.area(); } double visit(Circle& circle) override { return visit(const\_cast<const Circle&>(circle)); } double visit(const Rectangle& rect) override { return rect.area(); } double visit(Rectangle& rect) override { return visit(const\_cast<const Rectangle&>(rect)); } }; ╪º█î┘å ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘╛╪¬╪º┘å╪│█î┘ä ╪¬╪¡┘ê█î┘ä╪»┘ç┘å╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪▒╪º ╪»╪▒ ╪»╪º┘à┘å┘ç┘ç╪º█î ┌⌐╪º╪▒╪¿╪▒╪»█î ┘à╪¬┘å┘ê╪╣ ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ┘ü╪▒╪╡╪¬┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 110-116 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à╪╖╪º┘ä╪╣╪º╪¬ ╪«╪º╪╡ ╪»╪º┘à┘å┘ç ╪»╪▒ ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ORM╪î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î GUI╪î ╪▒┘ê╪┤╪┤┘å╪º╪│█î┘ç╪º█î ╪ó╪▓┘à╪º█î╪┤ ┘ê ╪º╪¬┘ê┘à╪º╪│█î┘ê┘å ╪º┘ä┌»┘ê█î ╪╖╪▒╪º╪¡█î ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 7. █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ ## 7.1 ╪¬╪╣╪º┘à┘ä Concept ┘ç╪º ┘ê ╪¿╪º╪▓╪¬╪º╪¿ ### 7.1.1 ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪¡╪»┘ê╪» ╪┤╪»┘ç ╪¬┘ê╪│╪╖ Concept █î┌⌐┘╛╪º╪▒┌å┌»█î concept ┘ç╪º█î C++20 ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘ç┘à╪º┘ü╪▓╪º█î█î┘ç╪º█î ┘é╪»╪▒╪¬┘à┘å╪»█î ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪º█î┘à┘å ┘å┘ê╪╣ ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å╪» [117]. Concept ┘ç╪º ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒┘ê┘å┌»╪▒█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» ┘ê ╪¿╪º ┘ç┘à ┘╛╪º█î┘ç ┘à╪¡┌⌐┘à█î ╪¿╪▒╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¬╪┤┌⌐█î┘ä ┘à█î╪»┘ç┘å╪»: cpp #include <concepts> #include <experimental/reflect> // Concept ┘ç╪º█î ╪ó┌»╪º┘ç ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> concept Reflectable = requires { std::meta::reflexpr(T); typename std::meta::data\_members\_t<std::meta::reflexpr(T)>; }; template<typename T> concept SerializableType = Reflectable<T> && requires { // ╪¿╪º█î╪» ╪º╪╣╪╢╪º█î ┘é╪º╪¿┘ä ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤╪» ┌⌐┘ç ╪«┘ê╪»╪┤╪º┘å ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘╛╪░█î╪▒ ╪¿╪º╪┤┘å╪» []<auto... Members>(std::index\_sequence<Members...>) { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); return (is\_serializable\_member<std::meta::get\_element\_v<Members, decltype(members)>>() && ...); }(std::make\_index\_sequence<std::meta::get\_size\_v<std::meta::data\_members\_of(std::meta::reflexpr(T))>>{}); }; template<auto Member> consteval bool is\_serializable\_member() { using member\_type = std::meta::get\_type\_t<Member>; if constexpr (std::is\_arithmetic\_v<member\_type>) { return true; } else if constexpr (std::is\_same\_v<member\_type, std::string>) { return true; } else if constexpr (Reflectable<member\_type>) { return SerializableType<member\_type>; } else { return false; } } // ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪¡╪»┘ê╪» ╪┤╪»┘ç ╪¬┘ê╪│╪╖ concept template<SerializableType T> std::string reflect\_serialize(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); std::ostringstream json; json << "{"; bool first = true; std::meta::template\_for<members>([&](auto member\_meta) { if (!first) json << ","; first = false; constexpr auto name = std::meta::get\_name\_v<member\_meta>; constexpr auto member\_ptr = std::meta::get\_pointer\_v<member\_meta>; json << "\"" << name << "\":"; serialize\_member\_value(json, obj.\*member\_ptr); }); json << "}"; return json.str(); } template<typename T> void serialize\_member\_value(std::ostringstream& json, const T& value) { if constexpr (std::is\_arithmetic\_v<T>) { json << value; } else if constexpr (std::is\_same\_v<T, std::string>) { json << "\"" << value << "\""; } else if constexpr (SerializableType<T>) { json << reflect\_serialize(value); } } ### 7.1.2 ╪¬╪╣╪º╪▒█î┘ü Concept ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪╣╪º╪▒█î┘ü concept ┘╛█î┌å█î╪»┘ç╪¬╪▒█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» ┌⌐┘ç ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ ╪▒╪º ╪¿┘ç ╪¼╪º█î ╪╡╪▒┘ü ╪▒╪º╪¿╪╖┘ç╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å┘å╪» [118]: cpp // Concept ┘ç╪º█î ╪│╪º╪«╪¬╪º╪▒█î ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> concept HasIdField = Reflectable<T> && requires { // ┘å┘ê╪╣ ╪¿╪º█î╪» ╪╣╪╢┘ê█î ╪¿╪º ┘å╪º┘à "id" ╪º╪▓ ┘å┘ê╪╣ ╪º┘å╪¬┌»╪▒╪º┘ä ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤╪» []() { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); return []<auto... Ms>(std::index\_sequence<Ms...>) { return ((std::meta::get\_name\_v<std::meta::get\_element\_v<Ms, decltype(members)>> == "id" && std::is\_integral\_v<std::meta::get\_type\_t<std::meta::get\_element\_v<Ms, decltype(members)>>>) || ...); }(std::make\_index\_sequence<std::meta::get\_size\_v<members>>{}); }(); }; template<typename T> concept DatabaseEntity = HasIdField<T> && requires { // ╪¿╪º█î╪» ╪¡╪º╪┤█î┘ç┘å┘ê█î╪│█î ┌⌐┘ä█î╪» ╪º╪╡┘ä█î ┘ê ┘å┌»╪º╪┤╪¬ ╪¼╪»┘ê┘ä ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤╪» []() { constexpr auto meta = std::meta::reflexpr(T); return std::meta::has\_attribute<entity\_table>(meta) && has\_primary\_key\_field(meta); }(); }; template<DatabaseEntity T> class repository { using id\_type = decltype(get\_id\_field\_type<T>()); public: std::optional<T> find\_by\_id(id\_type id) { constexpr auto table\_name = get\_table\_name<T>(); constexpr auto id\_column = get\_id\_column\_name<T>(); auto query = "SELECT \* FROM " + std::string(table\_name) + " WHERE " + std::string(id\_column) + " = ?"; return execute\_query<T>(query, id); } void save(const T& entity) { if constexpr (has\_auto\_increment\_id<T>()) { if (get\_id(entity) == 0) { insert(entity); } else { update(entity); } } else { upsert(entity); } } }; ### 7.1.3 ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º Concept ┘ç╪º ┘ê ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪▒┌⌐█î╪¿ ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘╛█î┌å█î╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» [119]: cpp // Concept ┘ç╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> concept ValidatedEntity = Reflectable<T> && requires { // ╪¬┘à╪º┘à ╪º╪╣╪╢╪º ╪¿╪º█î╪» ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘à┘å╪º╪│╪¿ ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤┘å╪» validate\_all\_members<T>(); }; template<typename T> consteval bool validate\_all\_members() { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); return []<auto... Ms>(std::index\_sequence<Ms...>) { return (validate\_member<std::meta::get\_element\_v<Ms, decltype(members)>>() && ...); }(std::make\_index\_sequence<std::meta::get\_size\_v<members>>{}); } template<auto Member> consteval bool validate\_member() { using member\_type = std::meta::get\_type\_t<Member>; // ╪º╪╣╪╢╪º█î ╪▒╪┤╪¬┘ç ╪¿╪º█î╪» ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪╖┘ê┘ä ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤┘å╪» if constexpr (std::is\_same\_v<member\_type, std::string>) { return std::meta::has\_attribute<max\_length>(Member) || std::meta::has\_attribute<regex\_pattern>(Member); } // ╪º╪╣╪╢╪º█î ╪╣╪»╪»█î ╪¿╪º█î╪» ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘à╪¡╪»┘ê╪»┘ç ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤┘å╪» else if constexpr (std::is\_arithmetic\_v<member\_type>) { return std::meta::has\_attribute<value\_range>(Member) || std::meta::has\_attribute<positive\_only>(Member); } // ╪│╪º█î╪▒ ╪º┘å┘ê╪º╪╣ ╪¿┘ç ╪╖┘ê╪▒ ┘╛█î╪┤┘ü╪▒╪╢ ┘à╪╣╪¬╪¿╪▒ ┘ç╪│╪¬┘å╪» else { return true; } } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä class $entity("users") $validated User { $primary\_key int id; $max\_length(100) $not\_empty std::string name; $range(0, 150) int age; $email\_format std::string email; $positive\_only double salary; // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪╖┘à█î┘å╪º┘å ┘à█î╪»┘ç╪» ┘ç┘à┘ç ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ┘à╪┤╪«╪╡ ╪┤╪»┘ç╪º┘å╪» }; // ┘à╪½╪º┘ä ┘å╪º┘à╪╣╪¬╪¿╪▒ - ╪«╪╖╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä class $entity("invalid") $validated BadUser { int id; std::string name; // ╪«╪╖╪º: ╪▒╪┤╪¬┘ç ╪¿╪»┘ê┘å ┘à╪¡╪»┘ê╪»█î╪¬ ╪╖┘ê┘ä int age; // ╪«╪╖╪º: ╪╣╪»╪»█î ╪¿╪»┘ê┘å ┘à╪¡╪»┘ê╪»█î╪¬ ┘à╪¡╪»┘ê╪»┘ç }; ## 7.2 Coroutine ┘ç╪º ┘ê ╪º┘ä┌»┘ê┘ç╪º█î Async ╪¿╪º╪▓╪¬╪º╪¿█î ### 7.2.1 ╪¬┘ê┘ä█î╪» Coroutine ╪▒╪º┘ç╪¿╪▒█î ╪┤╪»┘ç ╪¬┘ê╪│╪╖ ╪¿╪º╪▓╪¬╪º╪¿ Coroutine ┘ç╪º█î C++20 ╪¬╪▒┌⌐█î╪¿ ╪┤╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ API ┘ç╪º█î ┘å╪º┘ç┘à┌»╪º┘à ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓┘å╪» [120]: cpp #include <coroutine> #include <experimental/reflect> // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪╣┘à┘ä█î╪º╪¬ async ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ coroutine ┘ç╪º ┘ê ╪¿╪º╪▓╪¬╪º╪¿ constexpr void async\_service(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ┘à╪¬╪»┘ç╪º█î async ┘à╪¿╪¬┘å█î ╪¿╪▒ coroutine ╪¿╪▒╪º█î ╪¬┘à╪º┘à ┘à╪¬╪»┘ç╪º█î ╪╣┘à┘ê┘à█î constexpr auto methods = std::meta::public\_member\_functions\_of(target); std::meta::template\_for<methods>([&](auto method\_meta) { constexpr auto method\_name = std::meta::get\_name\_v<method\_meta>; constexpr auto return\_type = std::meta::get\_return\_type\_t<method\_meta>; constexpr auto parameters = std::meta::get\_parameters\_t<method\_meta>; // ╪¬┘ê┘ä█î╪» ┘å╪│╪«┘ç async ┘ç╪▒ ┘à╪¬╪» generate\_async\_method(target, method\_meta); }); // ╪¬┘ê┘ä█î╪» ╪▓█î╪▒╪│╪º╪«╪¬ coroutine generate\_coroutine\_infrastructure(target); } template<std::meta::info MethodMeta> constexpr void generate\_async\_method(std::meta::info target, MethodMeta method) { constexpr auto method\_name = std::meta::get\_name\_v<method>; constexpr auto return\_type = std::meta::get\_return\_type\_t<method>; std::string async\_method = "task<" + std::meta::get\_display\_name\_v<return\_type> + "> " + std::string(method\_name) + "\_async("; // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º constexpr auto params = std::meta::get\_parameters\_t<method>; std::meta::template\_for<params>([&](auto param\_meta) { // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘╛╪º╪▒╪º┘à╪¬╪▒ ╪¿┘ç ╪º┘à╪╢╪º█î ┘à╪¬╪» async }); async\_method += ") {\n"; async\_method += " co\_return co\_await async\_executor\_.schedule([this]() {\n"; async\_method += " return this->" + std::string(method\_name) + "("; // ╪º┘å╪¬┘é╪º┘ä ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º std::meta::template\_for<params>([&](auto param\_meta) { // ╪º┘å╪¬┘é╪º┘ä ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º ╪¿┘ç ┘à╪¬╪» ┘ç┘à┌»╪º┘à }); async\_method += ");\n });\n}\n"; std::meta::compiler.declare(target, async\_method); } // ┘å┘ê╪╣ task coroutine ╪¿╪▒╪º█î ╪╣┘à┘ä█î╪º╪¬ async template<typename T> class task { public: struct promise\_type { T value\_; std::exception\_ptr exception\_; task get\_return\_object() { return task{std::coroutine\_handle<promise\_type>::from\_promise(\*this)}; } std::suspend\_never initial\_suspend() { return {}; } std::suspend\_never final\_suspend() noexcept { return {}; } void return\_value(T value) { value\_ = std::move(value); } void unhandled\_exception() { exception\_ = std::current\_exception(); } }; private: std::coroutine\_handle<promise\_type> handle\_; public: explicit task(std::coroutine\_handle<promise\_type> handle) : handle\_(handle) {} ~task() { if (handle\_) { handle\_.destroy(); } } T get() { if (handle\_.promise().exception\_) { std::rethrow\_exception(handle\_.promise().exception\_); } return std::move(handle\_.promise().value\_); } bool ready() const { return handle\_.done(); } }; // ┘à╪½╪º┘ä ╪│╪▒┘ê█î╪│ ╪¿╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ async class $async\_service DataService { database::connection db\_; public: User get\_user(int id) { return db\_.query<User>("SELECT \* FROM users WHERE id = ?", id); } std::vector<User> get\_users\_by\_department(const std::string& dept) { return db\_.query<std::vector<User>>( "SELECT \* FROM users WHERE department = ?", dept); } void update\_user(const User& user) { db\_.execute("UPDATE users SET name = ?, age = ? WHERE id = ?", user.name, user.age, user.id); } // ╪¿┘ç ╪╖┘ê╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪»: // task<User> get\_user\_async(int id); // task<std::vector<User>> get\_users\_by\_department\_async(const std::string& dept); // task<void> update\_user\_async(const User& user); }; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º API ╪«┘ê╪»┌⌐╪º╪▒ async async\_task<void> process\_users() { DataService service; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¬╪»┘ç╪º█î async ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç auto user = co\_await service.get\_user\_async(123); auto dept\_users = co\_await service.get\_users\_by\_department\_async("Engineering"); user.salary \*= 1.1; // ╪º┘ü╪▓╪º█î╪┤ 10% ╪¡┘é┘ê┘é co\_await service.update\_user\_async(user); } ### 7.2.2 ╪¼╪▒█î╪º┘å ╪▒┘ê█î╪»╪º╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪¼╪▒█î╪º┘å┘ç╪º█î ╪▒┘ê█î╪»╪º╪» ┘ê╪º┌⌐┘å╪┤█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î╪│╪º╪▓╪» [121]: cpp // ╪¼╪▒█î╪º┘å ╪▒┘ê█î╪»╪º╪» ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê coroutine ┘ç╪º template<typename T> concept EventStreamable = Reflectable<T> && requires { std::meta::has\_metaclass<observable>(T); }; template<EventStreamable T> class event\_stream { public: using value\_type = T; template<auto Member> auto observe\_member() -> async\_generator<std::meta::get\_type\_t<Member>> { constexpr auto member\_name = std::meta::get\_name\_v<Member>; while (true) { auto change\_event = co\_await wait\_for\_change(member\_name); auto new\_value = source\_.\*std::meta::get\_pointer\_v<Member>; co\_yield new\_value; } } auto observe\_all\_changes() -> async\_generator<property\_change\_event> { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); std::meta::template\_for<members>([&](auto member\_meta) { setup\_member\_observer(member\_meta); }); while (true) { auto event = co\_await wait\_for\_any\_change(); co\_yield event; } } private: T\* source\_; std::unordered\_map<std::string, std::queue<std::any>> change\_queues\_; template<auto Member> void setup\_member\_observer(Member member) { constexpr auto member\_name = std::meta::get\_name\_v<member>; source\_->add\_observer<std::meta::get\_type\_t<member>>( member\_name, [this](const auto& new\_value) { change\_queues\_[member\_name].push(std::make\_any(new\_value)); notify\_change(member\_name); } ); } }; // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç async\_task<void> monitor\_stock\_prices() { StockPrice stock{"AAPL", 150.0, 1000000}; event\_stream<StockPrice> stream(&stock); // ┘å╪╕╪º╪▒╪¬ ╪¿╪▒ ╪¬╪║█î█î╪▒╪º╪¬ ╪╣╪╢┘ê ╪«╪º╪╡ auto price\_stream = stream.observe\_member<&StockPrice::price>(); auto volume\_stream = stream.observe\_member<&StockPrice::volume>(); // ┘╛╪▒╪»╪º╪▓╪┤ ┘å╪º┘ç┘à┌»╪º┘à ╪¬╪║█î█î╪▒╪º╪¬ ┘é█î┘à╪¬ while (auto price = co\_await price\_stream.next()) { if (price > 160.0) { std::cout << "Price alert: $" << price << std::endl; } } } ## 7.3 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪│█î╪│╪¬┘à Module ┘ç╪º ### 7.3.1 ╪¿╪º╪▓╪¬╪º╪¿ ╪ó┌»╪º┘ç ╪¿┘ç Module Module ┘ç╪º█î C++20 ┘å█î╪º╪▓ ╪¿┘ç ╪¬┘ê╪¼┘ç ┘ê█î┌ÿ┘ç ╪¿╪▒╪º█î ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪º╪▒┘å╪» [122]: cpp // ╪▒╪º╪¿╪╖ module ╪¿╪º ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪¬╪º╪¿ export module data\_models; import std.core; import std.reflection; // ╪╡╪º╪»╪▒╪º╪¬ ╪º┘å┘ê╪º╪╣ ╪¿╪º╪▓╪¬╪º╪¿█î ╪¿╪º ╪»█î╪» module export template<typename T> concept ModuleReflectable = requires { std::meta::reflexpr(T); std::meta::is\_exported\_v<std::meta::reflexpr(T)>; }; export class $serializable $entity("users") User { int id; std::string name; std::string email; // ┘ü╪▒╪º╪»╪º╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ┘å┘ê╪╣ ╪╡╪º╪»╪▒ ┘à█î╪┤┘ê╪» }; export class $serializable Product { int product\_id; std::string name; double price; }; // ╪╡╪º╪»╪▒╪º╪¬ ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪º┘å┘ê╪º╪╣ module export template<ModuleReflectable T> std::string serialize\_module\_type(const T& obj) { constexpr auto meta = std::meta::reflexpr(T); static\_assert(std::meta::is\_exported\_v<meta>, "Type must be exported for cross-module reflection"); return reflect\_serialize\_impl(obj, meta); } // ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪«╪╡┘ê╪╡█î module namespace detail { template<std::meta::info TypeMeta> constexpr bool is\_module\_exportable() { return std::meta::is\_public\_v<TypeMeta> && std::meta::has\_export\_declaration\_v<TypeMeta>; } } ### 7.3.2 ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿█î┘å Module ┘ç╪º ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪º█î╪» ╪¿┘ç ╪»╪▒╪│╪¬█î ╪»╪▒ ╪│╪▒╪º╪│╪▒ ┘à╪▒╪▓┘ç╪º█î module ┌⌐╪º╪▒ ┌⌐┘å┘å╪» [123]: cpp // ╪¬╪╣╪º╪▒█î┘ü ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪»╪▒ module ╪¼╪»╪º┌»╪º┘å┘ç export module metaclasses.serialization; import std.core; import std.reflection; // ╪╡╪º╪»╪▒╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ export constexpr void serializable(std::meta::info target, serialization\_format format = json) { // ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ╪│╪º╪▓┌»╪º╪▒█î ╪¿█î┘å module ┘ç╪º std::meta::compiler.require( std::meta::is\_module\_exported\_v<target> || std::meta::is\_module\_internal\_v<target>, "serializable can only be applied to exported or internal types" ); generate\_serialization\_methods(target, format); } export constexpr void entity(std::meta::info target, std::string\_view table\_name = "") { // ┘╛╪┤╪¬█î╪¿╪º┘å█î entity ╪¿█î┘å module ┘ç╪º validate\_cross\_module\_entity(target); generate\_entity\_methods(target, table\_name); } // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿█î┘å module ┘ç╪º constexpr void validate\_cross\_module\_entity(std::meta::info target) { // ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ╪»╪│╪¬╪▒╪│█î ┘ç┘à┘ç ╪º┘å┘ê╪º╪╣ ┘ê╪º╪¿╪│╪¬┘ç constexpr auto members = std::meta::data\_members\_of(target); std::meta::template\_for<members>([&](auto member\_meta) { constexpr auto member\_type = std::meta::get\_type\_t<member\_meta>; static\_assert( std::meta::is\_module\_accessible\_v<member\_type>, "All entity member types must be accessible across modules" ); }); } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪»╪▒ module ┌⌐┘ä╪º█î┘å╪¬ module client; import data\_models; import metaclasses.serialization; void process\_data() { User user{1, "Alice", "alice@example.com"}; // ╪¿╪º╪▓╪¬╪º╪¿ ╪¿█î┘å module ┘ç╪º ╪¿╪»┘ê┘å ┘à╪┤┌⌐┘ä ┌⌐╪º╪▒ ┘à█î┌⌐┘å╪» std::string json = user.to\_json(); User restored = User::from\_json(json); } ## 7.4 ┘ü╪▒╪╡╪¬┘ç╪º█î ╪¬┘é┘ê█î╪¬ ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç Ranges ### 7.4.1 ╪º┘ä┌»┘ê╪▒█î╪¬┘à┘ç╪º█î Range ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ Range ┘ç╪º█î C++20 ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ┘╛╪▒╪»╪º╪▓╪┤ ╪«┘ê╪»┌⌐╪º╪▒ ╪»╪º╪»┘ç ╪¬┘é┘ê█î╪¬ ╪┤┘ê┘å╪» [124]: cpp #include <ranges> #include <experimental/reflect> // ╪¬╪¿╪»█î┘ä┘ç╪º█î range ╪ó┌»╪º┘ç ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ namespace ranges::reflection { template<typename T> concept ReflectableRange = std::ranges::range<T> && Reflectable<std::ranges::range\_value\_t<T>>; // ╪º╪│╪¬╪«╪▒╪º╪¼ ╪«┘ê╪»┌⌐╪º╪▒ ╪╣╪╢┘ê template<auto Member> struct extract\_member { template<typename T> constexpr auto operator()(const T& obj) const { return obj.\*std::meta::get\_pointer\_v<Member>; } }; template<auto Member> constexpr auto extract = extract\_member<Member>{}; // ┘ü█î┘ä╪¬╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒ ╪º╪│╪º╪│ ┘à┘é╪º╪»█î╪▒ ╪╣╪╢┘ê template<auto Member, typename Predicate> struct filter\_by\_member { Predicate pred; template<typename T> constexpr bool operator()(const T& obj) const { return pred(obj.\*std::meta::get\_pointer\_v<Member>); } }; template<auto Member, typename Predicate> constexpr auto filter\_by = [](Predicate pred) { return filter\_by\_member<Member, Predicate>{pred}; }; // ┌»╪▒┘ê┘ç╪¿┘å╪»█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒ ╪º╪│╪º╪│ ┘à┘é╪º╪»█î╪▒ ╪╣╪╢┘ê template<auto Member> struct group\_by\_member { template<ReflectableRange Range> auto operator()(Range&& range) const { using key\_type = std::meta::get\_type\_t<Member>; using value\_type = std::ranges::range\_value\_t<Range>; std::map<key\_type, std::vector<value\_type>> groups; for (const auto& item : range) { auto key = item.\*std::meta::get\_pointer\_v<Member>; groups[key].push\_back(item); } return groups; } }; template<auto Member> constexpr auto group\_by = group\_by\_member<Member>{}; // ╪¬╪¼┘à█î╪╣ ╪«┘ê╪»┌⌐╪º╪▒ template<auto Member, typename BinaryOp> struct aggregate\_member { BinaryOp op; template<ReflectableRange Range> auto operator()(Range&& range) const { using member\_type = std::meta::get\_type\_t<Member>; if (std::ranges::empty(range)) { return member\_type{}; } auto first = std::ranges::begin(range); auto init = (\*first).\*std::meta::get\_pointer\_v<Member>; return std::ranges::fold\_left( range | std::views::drop(1) | std::views::transform(extract<Member>), init, op ); } }; template<auto Member, typename BinaryOp> constexpr auto aggregate = [](BinaryOp op) { return aggregate\_member<Member, BinaryOp>{op}; }; } // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º range ┘ç╪º█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ struct Employee { int id; std::string name; std::string department; double salary; int years\_experience; }; void demonstrate\_reflection\_ranges() { std::vector<Employee> employees = { {1, "Alice", "Engineering", 95000, 5}, {2, "Bob", "Engineering", 87000, 3}, {3, "Carol", "Marketing", 78000, 7}, {4, "David", "Engineering", 102000, 8}, {5, "Eve", "Marketing", 83000, 4} }; using namespace ranges::reflection; // ╪º╪│╪¬╪«╪▒╪º╪¼ ╪¬┘à╪º┘à ╪¡┘é┘ê┘é┘ç╪º auto salaries = employees | std::views::transform(extract<&Employee::salary>) | std::ranges::to<std::vector>(); // ┘ü█î┘ä╪¬╪▒ ┌⌐╪│╪º┘å█î ╪¿╪º ╪»╪▒╪ó┘à╪» ╪¿╪º┘ä╪º auto high\_earners = employees | std::views::filter(filter\_by<&Employee::salary>([](double s) { return s > 90000; })) | std::ranges::to<std::vector>(); // ┌»╪▒┘ê┘ç╪¿┘å╪»█î ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪«╪┤ auto by\_department = employees | group\_by<&Employee::department>; // ┘à╪¡╪º╪│╪¿┘ç ┌⌐┘ä ╪¡┘é┘ê┘é ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪«╪┤ for (const auto& [dept, emps] : by\_department) { auto total\_salary = emps | aggregate<&Employee::salary>(std::plus{}); std::cout << dept << ": $" << total\_salary << std::endl; } // ┘à█î╪º┘å┌»█î┘å ╪│╪º┘ä┘ç╪º█î ╪¬╪¼╪▒╪¿┘ç ╪¿╪▒╪º█î ┘à┘ç┘å╪»╪│╪º┘å auto engineers = by\_department["Engineering"]; auto avg\_experience = static\_cast<double>( engineers | aggregate<&Employee::years\_experience>(std::plus{}) ) / engineers.size(); std::cout << "Average engineering experience: " << avg\_experience << " years" << std::endl; } ### 7.4.2 ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪ó╪»╪º┘╛╪¬┘ê╪▒ Range ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å╪» ╪ó╪»╪º┘╛╪¬┘ê╪▒┘ç╪º█î range ╪│┘ü╪º╪▒╪┤█î ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ ╪¬┘ê┘ä█î╪» ┌⌐┘å╪» [125]: cpp // ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪ó╪»╪º┘╛╪¬┘ê╪▒ range ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> class reflected\_range\_adapters { static\_assert(Reflectable<T>); public: // ╪¬┘ê┘ä█î╪» view ┘ç╪º█î ┘à┘é╪º█î╪│┘ç ╪╣╪╢┘ê ╪¿┘ç ╪╣╪╢┘ê template<auto Member> static auto equal\_to(const std::meta::get\_type\_t<Member>& value) { return std::views::filter([value](const T& obj) { return obj.\*std::meta::get\_pointer\_v<Member> == value; }); } template<auto Member> static auto greater\_than(const std::meta::get\_type\_t<Member>& value) { return std::views::filter([value](const T& obj) { return obj.\*std::meta::get\_pointer\_v<Member> > value; }); } // ╪¬┘ê┘ä█î╪» view ┘ç╪º█î ┘à╪▒╪¬╪¿╪│╪º╪▓█î template<auto Member> static auto sort\_by\_ascending() { return [](auto&& range) { auto sorted = range | std::ranges::to<std::vector>(); std::ranges::sort(sorted, [](const T& a, const T& b) { return (a.\*std::meta::get\_pointer\_v<Member>) < (b.\*std::meta::get\_pointer\_v<Member>); }); return sorted; }; } // ╪¬┘ê┘ä█î╪» view ┘ç╪º█î projection ╪¿╪▒╪º█î ╪¬┘à╪º┘à ╪º╪╣╪╢╪º static auto project\_all\_members() { constexpr auto meta = std::meta::reflexpr(T); constexpr auto members = std::meta::data\_members\_of(meta); return [](const T& obj) { return std::make\_tuple( obj.\*std::meta::get\_pointer\_v< std::meta::get\_element\_v<0, decltype(members)>>.. ); }; } }; // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç void demonstrate\_automatic\_adapters() { std::vector<Employee> employees = /\* ... \*/; using adapters = reflected\_range\_adapters<Employee>; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪ó╪»╪º┘╛╪¬┘ê╪▒┘ç╪º█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç auto high\_salary = employees | adapters::greater\_than<&Employee::salary>(90000) | std::ranges::to<std::vector>(); auto engineers = employees | adapters::equal\_to<&Employee::department>("Engineering") | std::ranges::to<std::vector>(); auto sorted\_by\_experience = employees | adapters::sort\_by\_ascending<&Employee::years\_experience>() | std::ranges::to<std::vector>(); } ## 7.5 ╪º┘ä┌»┘ê┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ### 7.5.1 ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ╪ó┌»╪º┘ç ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪º ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘é┘ê█î╪¬ ╪┤┘ê┘å╪» [126]: cpp // vector ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ╪╣┘à┘ä█î╪º╪¬ ╪«┘ê╪»┌⌐╪º╪▒ template<Reflectable T> class reflected\_vector : public std::vector<T> { using base = std::vector<T>; public: using base::base; // ╪º╪▒╪½╪¿╪▒█î ╪│╪º╪▓┘å╪»┘ç┘ç╪º // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ┌⌐┘ä ┌⌐╪º┘å╪¬█î┘å╪▒ std::string to\_json() const { std::ostringstream json; json << "["; bool first = true; for (const auto& item : \*this) { if (!first) json << ","; first = false; json << reflect\_serialize(item); } json << "]"; return json.str(); } // ┘ü█î┘ä╪¬╪▒ ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê╪│╪╖ ┘ç╪▒ ╪╣╪╢┘ê template<auto Member, typename Predicate> reflected\_vector filter\_by(Predicate pred) const { reflected\_vector result; std::ranges::copy\_if(\*this, std::back\_inserter(result), [pred](const T& item) { return pred(item.\*std::meta::get\_pointer\_v<Member>); } ); return result; } // ┌»╪▒┘ê┘ç╪¿┘å╪»█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¬┘ê╪│╪╖ ┘ç╪▒ ╪╣╪╢┘ê template<auto Member> auto group\_by() const { using key\_type = std::meta::get\_type\_t<Member>; std::map<key\_type, reflected\_vector> groups; for (const auto& item : \*this) { auto key = item.\*std::meta::get\_pointer\_v<Member>; groups[key].push\_back(item); } return groups; } // ╪º╪│╪¬╪«╪▒╪º╪¼ ╪«┘ê╪»┌⌐╪º╪▒ ╪╣╪╢┘ê template<auto Member> auto extract\_member() const { using member\_type = std::meta::get\_type\_t<Member>; std::vector<member\_type> result; std::ranges::transform(\*this, std::back\_inserter(result), [](const T& item) { return item.\*std::meta::get\_pointer\_v<Member>; } ); return result; } // ╪¼╪│╪¬╪¼┘ê█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪º ┘à╪╣█î╪º╪▒┘ç╪º█î ╪╣╪╢┘ê template<auto Member> auto find\_by(const std::meta::get\_type\_t<Member>& value) const { return std::ranges::find\_if(\*this, [value](const T& item) { return item.\*std::meta::get\_pointer\_v<Member> == value; }); } }; // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç void demonstrate\_reflected\_containers() { reflected\_vector<Employee> employees = { {1, "Alice", "Engineering", 95000, 5}, {2, "Bob", "Engineering", 87000, 3}, {3, "Carol", "Marketing", 78000, 7} }; // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪╣┘à┘ä█î╪º╪¬ ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ auto json\_data = employees.to\_json(); auto high\_earners = employees.filter\_by<&Employee::salary>( [](double salary) { return salary > 90000; } ); auto by\_department = employees.group\_by<&Employee::department>(); auto salaries = employees.extract\_member<&Employee::salary>(); auto alice = employees.find\_by<&Employee::name>("Alice"); } ╪º█î┘å ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ █î┌⌐┘╛╪º╪▒┌å┌»█î ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌å┌»┘ê┘å┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪¿╪º ╪│╪º█î╪▒ ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ ┘ç┘à╪º┘ü╪▓╪º█î█î ┘à█î┌⌐┘å┘å╪» ╪¬╪º ┘╛╪º╪▒╪º╪»╪º█î┘à┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪»╪▒╪¬┘à┘å╪» ╪º█î╪¼╪º╪» ┌⌐┘å┘å╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ┌å╪º┘ä╪┤┘ç╪º ┘ê ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪º█î┘å ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 117-126 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à╪╖╪º┘ä╪╣╪º╪¬ █î┌⌐┘╛╪º╪▒┌å┌»█î concept-╪¿╪º╪▓╪¬╪º╪¿╪î ╪º┘ä┌»┘ê┘ç╪º█î ╪¬┘é┘ê█î╪¬ coroutine╪î ╪│╪º╪▓┌»╪º╪▒█î ╪│█î╪│╪¬┘à module╪î ┘╛╪│┘ê┘å╪»┘ç╪º█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ranges ┘ê ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 8. ┌å╪º┘ä╪┤┘ç╪º ┘ê ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º ## 8.1 ┘╛█î┌å█î╪»┌»█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ### 8.1.1 ┌å╪º┘ä╪┤┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î Frontend ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪»╪▒ frontend ┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┌å╪º┘ä╪┤┘ç╪º█î ┘ü┘å█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [127]. ╪¿╪▒╪«┘ä╪º┘ü ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪│┘å╪¬█î ╪▓╪¿╪º┘å ┌⌐┘ç ╪▒┘ê█î ┘å╪¡┘ê ┌⌐╪º┘à┘ä╪º┘ï ╪¬╪╣╪▒█î┘ü ╪┤╪»┘ç ╪╣┘à┘ä ┘à█î┌⌐┘å┘å╪»╪î ╪¿╪º╪▓╪¬╪º╪¿ ┘å█î╪º╪▓ ╪¿┘ç █î┌⌐┘╛╪º╪▒┌å┌»█î ╪╣┘à█î┘é ╪¿╪º ╪│█î╪│╪¬┘à ┘å┘ê╪╣ ╪»╪º╪«┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ê ┘à╪▒╪º╪¡┘ä ╪¬╪¡┘ä█î┘ä ┘à╪╣┘å╪º█î█î ╪»╪º╪▒╪». **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪¼╪»┘ê┘ä ┘å┘à╪º╪»:** cpp // ┘à┘ä╪º╪¡╪╕╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ namespace compiler::reflection { // ╪¿╪º╪▓╪¬╪º╪¿ ┘å█î╪º╪▓ ╪¿┘ç ╪░╪«█î╪▒┘ç╪│╪º╪▓█î ┘à╪»╪º┘ê┘à meta-object ╪»╪º╪▒╪» class meta\_object\_registry { // ╪¿╪º█î╪» ╪»╪▒ ┘à╪▒╪º╪¡┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º┘é█î ╪¿┘à╪º┘å╪» std::unordered\_map<type\_id, meta\_info> type\_registry\_; std::unordered\_map<symbol\_id, meta\_info> symbol\_registry\_; // ┌å╪º┘ä╪┤┘ç╪º█î ╪½╪¿╪º╪¬ ╪¿█î┘å ┘ê╪º╪¡╪» ╪¬╪▒╪¼┘à┘ç std::unordered\_map<module\_id, std::vector<exported\_meta\_info>> module\_exports\_; public: // ╪º█î┘à┘å█î thread ╪¿╪▒╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ┘à┘ê╪º╪▓█î ┘à┘ê╪▒╪» ┘å█î╪º╪▓ ╪º╪│╪¬ meta\_info get\_type\_info(type\_id id) const; // ╪¿╪º█î╪» ╪▓┘à█î┘å┘ç┘ç╪º█î instantiation ┘é╪º┘ä╪¿ ╪▒╪º ┘à╪»█î╪▒█î╪¬ ┌⌐┘å╪» meta\_info instantiate\_template\_meta(template\_id id, const instantiation\_args& args); // ╪▒╪»█î╪º╪¿█î ┘ê╪º╪¿╪│╪¬┌»█î ┘╛█î┌å█î╪»┘ç ╪¿╪▒╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î void register\_meta\_dependency(meta\_info dependent, meta\_info dependency); }; // ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º█î╪» ╪¿╪º ╪º╪▒╪▓█î╪º╪¿█î ╪½╪º╪¿╪¬ █î┌⌐┘╛╪º╪▒┌å┘ç ╪┤┘ê┘å╪» class constexpr\_reflection\_evaluator { // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪╖█î ╪º╪▒╪▓█î╪º╪¿█î ╪½╪º╪¿╪¬ constexpr\_value evaluate\_reflection\_query(const reflection\_expr& expr); // expansion ╪¿╪│╪¬┘ç ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ std::vector<constexpr\_value> expand\_reflected\_pack(const pack\_expr& expr); // ╪¼╪▒█î╪º┘å ╪»╪º╪»┘ç ╪¿█î┘å ┘à╪▒╪º╪¡┘ä: constexpr ╪¿┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» void register\_code\_generation\_request(const metaclass\_application& app); }; } **┘╛█î┌å█î╪»┌»█î Instantiation ┘é╪º┘ä╪¿:** ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪╖┘ê╪▒ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î instantiation ┘é╪º┘ä╪¿ ╪▒╪º ┘╛█î┌å█î╪»┘ç ┘à█î┌⌐┘å╪»╪î ┌å╪▒╪º ┌⌐┘ç meta-object ┘ç╪º ╪¿╪º█î╪» ╪╖█î instantiation ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿╪º╪┤┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪»┘ä two-phase lookup ╪▒╪º ╪▒╪╣╪º█î╪¬ ┌⌐┘å┘å╪» [128]: cpp // ┌å╪º┘ä╪┤┘ç╪º█î instantiation ┘é╪º┘ä╪¿ template<typename T> void problematic\_template() { // Meta-object ╪¿╪º█î╪» ╪╖█î instantiation ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿╪º╪┤╪» constexpr auto meta = std::meta::reflexpr(T); // ╪º┘à╪º T ┘à┘à┌⌐┘å ╪º╪│╪¬ ╪»╪▒ ┘à╪▒╪¡┘ä┘ç ╪º┘ê┘ä ┌⌐╪º┘à┘ä ┘å╪¿╪º╪┤╪» constexpr auto members = std::meta::data\_members\_of(meta); // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪╖█î instantiation std::meta::template\_for<members>([](auto member) { // ┘ç╪▒ ╪¬┌⌐╪▒╪º╪▒ ┘å█î╪º╪▓ ╪¿┘ç ┘ê╪╢╪╣█î╪¬ ╪¬╪º╪▓┘ç ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪»╪º╪▒╪» generate\_code\_for\_member(member); }); } // ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¿╪º█î╪» ┘à╪»█î╪▒█î╪¬ ┌⌐┘å╪»: // 1. ╪º█î╪¼╪º╪» ╪¬╪╣┘ê█î┘é ╪┤╪»┘ç meta-object // 2. ╪¬╪«╪╡╪╡ ┘é╪º┘ä╪¿ ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ // 3. SFINAE ╪¿╪º ┘à╪¡┘à┘ê┘ä┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ // 4. ╪º╪▒╪▓█î╪º╪¿█î concept ╪¿╪º ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ### 8.1.2 ┌å╪º┘ä╪┤┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» Backend ┘╛█î╪º╪»┘ç╪│╪º╪▓█î backend ╪¿╪º ┌å╪º┘ä╪┤┘ç╪º█î ┘à┘å╪¡╪╡╪▒╪¿┘ü╪▒╪»█î ╪»╪▒ ╪¬┘ê┘ä█î╪» ┌⌐╪» ┌⌐╪º╪▒╪º ╪º╪▓ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘ê╪º╪¼┘ç ╪º╪│╪¬ [129]: cpp // ┘╛█î┌å█î╪»┌»█î ╪¬┘ê┘ä█î╪» ┌⌐╪» backend namespace compiler::codegen { class metaclass\_code\_generator { // ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º█î╪» ╪¿╪»┘ê┘å ┘à╪┤┌⌐┘ä ╪¿╪º ┌⌐╪» ┘à┘ê╪¼┘ê╪» █î┌⌐┘╛╪º╪▒┌å┘ç ╪┤┘ê╪» llvm::Value\* generate\_reflection\_query(const reflection\_query& query, llvm::IRBuilder<>& builder); // Instantiation ┘é╪º┘ä╪¿ ┘à█î╪¬┘ê╪º┘å╪» ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▒╪º ┘ü╪╣╪º┘ä ┌⌐┘å╪» void handle\_deferred\_generation(const deferred\_generation\_request& request); // ┘ç┘à╪º┘ç┘å┌»█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿█î┘å module ┘ç╪º void coordinate\_cross\_module\_generation(const module\_interface& interface); // ╪¡┘ü╪╕ ╪º╪╖┘ä╪º╪╣╪º╪¬ debug ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç void preserve\_debug\_info(const generated\_code\_section& section, const source\_location& original\_location); }; // ┌å╪º┘ä╪┤┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪¿╪º ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç class reflection\_optimizer { // ╪¡╪░┘ü ┌⌐╪» ┘à╪▒╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ bool is\_reflection\_generated\_code\_reachable(const llvm::Function& func); // ╪¬╪╡┘à█î┘à╪º╪¬ inlining ╪¿╪▒╪º█î ┘à╪¬╪»┘ç╪º█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç bool should\_inline\_generated\_method(const method\_info& method); // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪¿█î┘å ╪¬╪º╪¿╪╣ ╪¿╪º ┘à╪▒╪▓┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ void optimize\_across\_reflection\_boundaries(llvm::Module& module); }; } ### 8.1.3 ┘à┘ä╪º╪¡╪╕╪º╪¬ ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î ╪¿╪º╪▓╪¬╪º╪¿ ┌å╪º┘ä╪┤┘ç╪º█î ╪«╪º╪╡█î ╪¿╪▒╪º█î ╪│█î╪│╪¬┘à┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å╪» [130]: cpp // ╪▒╪»█î╪º╪¿█î ┘ê╪º╪¿╪│╪¬┌»█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î namespace build\_system { class reflection\_dependency\_tracker { // ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘╛█î┌å█î╪»┘ç╪¬╪▒ ╪º╪▓ ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ╪│┘å╪¬█î ┘ç╪│╪¬┘å╪» struct reflection\_dependency { source\_file dependent\_file; type\_identifier reflected\_type; std::vector<member\_identifier> accessed\_members; metaclass\_set applied\_metaclasses; // ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ╪º┘å╪¬┘é╪º┘ä█î ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ std::vector<reflection\_dependency> transitive\_deps; }; // ╪¬╪¡┘ä█î┘ä ╪¬╪ú╪½█î╪▒ ╪¬╪║█î█î╪▒ ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ std::vector<source\_file> compute\_affected\_files( const std::vector<changed\_file>& changes) { std::vector<source\_file> affected; for (const auto& change : changes) { // ┘ê╪º╪¿╪│╪¬┘ç┘ç╪º█î ┘à╪│╪¬┘é█î┘à auto direct = get\_direct\_dependents(change); affected.insert(affected.end(), direct.begin(), direct.end()); // ┘ê╪º╪¿╪│╪¬┘ç┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ auto reflection\_deps = get\_reflection\_dependents(change); affected.insert(affected.end(), reflection\_deps.begin(), reflection\_deps.end()); // ┘ê╪º╪¿╪│╪¬┘ç┘ç╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ auto generated\_deps = get\_generated\_code\_dependents(change); affected.insert(affected.end(), generated\_deps.begin(), generated\_deps.end()); } return affected; } private: // ╪¬╪¡┘ä█î┘ä ┘╛█î┌å█î╪»┘ç ╪¿╪▒╪º█î ╪¬╪║█î█î╪▒╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘ê╪▒╪» ┘å█î╪º╪▓ ╪º╪│╪¬ std::vector<source\_file> get\_reflection\_dependents(const changed\_file& file); std::vector<source\_file> get\_generated\_code\_dependents(const changed\_file& file); }; } ## 8.2 ╪»█î╪¿╪º┌» ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿█î ### 8.2.1 ┌å╪º┘ä╪┤┘ç╪º█î ┘å┌»╪º╪┤╪¬ ┌⌐╪» ┘à┘å╪¿╪╣ ╪»█î╪¿╪º┌» ┌⌐╪»█î ┌⌐┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┌»╪│╪¬╪▒╪»┘ç ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å╪» ┌å╪º┘ä╪┤┘ç╪º█î ┘à┘å╪¡╪╡╪▒╪¿┘ü╪▒╪»█î ╪¿╪▒╪º█î ┘å┘ê█î╪│┘å╪»┌»╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ê ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┌⌐╪º╪▒╪¿╪▒╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [131]: cpp // ╪▓█î╪▒╪│╪º╪«╪¬ ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪»█î╪¿╪º┌» namespace debugging { // ┘å┌»╪º╪┤╪¬ ┘à┘å╪¿╪╣ ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç class reflection\_debug\_info { // ┘å┌»╪º╪┤╪¬ ┘à┌⌐╪º┘å┘ç╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒┌»╪┤╪¬ ╪¿┘ç ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ struct code\_provenance { source\_location metaclass\_application\_site; source\_location original\_type\_definition; std::string generation\_context; std::vector<reflection\_operation> generation\_steps; }; // ╪º╪╖┘ä╪º╪╣╪º╪¬ debug ╪¿╪▒╪º█î ╪º╪╣╪╢╪º█î ╪¿╪º╪▓╪¬╪º╪¿█î struct reflected\_member\_debug\_info { std::string original\_name; source\_location definition\_site; type\_info original\_type; std::vector<attribute> applied\_attributes; }; public: // ╪º╪▒╪º╪ª┘ç stack trace ┘à╪╣┘å╪º╪»╪º╪▒ ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç std::vector<stack\_frame> get\_enhanced\_stack\_trace( const std::vector<raw\_stack\_frame>& raw\_frames) { std::vector<stack\_frame> enhanced; for (const auto& frame : raw\_frames) { if (is\_generated\_code(frame.address)) { // ┘å┌»╪º╪┤╪¬ ╪¿╪▒┌»╪┤╪¬ ╪¿┘ç ┘à┘å╪¿╪╣ ╪º╪╡┘ä█î auto provenance = get\_code\_provenance(frame.address); enhanced.emplace\_back(create\_enhanced\_frame(frame, provenance)); } else { enhanced.push\_back(frame); } } return enhanced; } // ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¬┘å╪╕█î┘à breakpoint ╪»╪▒ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç std::vector<debug\_location> resolve\_breakpoint\_locations( const source\_location& user\_specified\_location) { std::vector<debug\_location> locations; // ┘à┌⌐╪º┘å ┘à╪│╪¬┘é█î┘à locations.push\_back(user\_specified\_location); // ┘à┌⌐╪º┘å┘ç╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ┌⌐┘ç ╪¿╪º ╪º█î┘å ┘à┘å╪¿╪╣ ┘à╪╖╪º╪¿┘é╪¬ ╪»╪º╪▒┘å╪» auto generated = find\_generated\_locations(user\_specified\_location); locations.insert(locations.end(), generated.begin(), generated.end()); return locations; } }; // █î┌⌐┘╛╪º╪▒┌å┌»█î debugger ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ class reflection\_debugger\_support { public: // ╪¿╪º╪▓╪▒╪│█î meta-object ┘ç╪º ╪»╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪▒╪º█î ╪»█î╪¿╪º┌» std::string format\_meta\_object(const std::meta::info& meta\_obj) { // ┘ü╪▒┘à╪¬ ╪º╪╖┘ä╪º╪╣╪º╪¬ meta-object ╪¿╪▒╪º█î ┘å┘à╪º█î╪┤ debugger std::ostringstream result; result << "Meta-object type: " << get\_meta\_object\_type(meta\_obj) << "\n"; result << "Represented entity: " << get\_represented\_entity\_name(meta\_obj) << "\n"; if (is\_type\_meta\_object(meta\_obj)) { format\_type\_meta\_object(result, meta\_obj); } else if (is\_member\_meta\_object(meta\_obj)) { format\_member\_meta\_object(result, meta\_obj); } return result.str(); } // ╪¿╪º╪▓╪▒╪│█î ┘à╪¬╪║█î╪▒ ╪¿╪º ╪▓┘à█î┘å┘ç ╪¿╪º╪▓╪¬╪º╪¿ inspection\_result inspect\_reflected\_variable( const variable\_reference& var\_ref) { if (!has\_reflection\_type(var\_ref)) { return standard\_inspection(var\_ref); } // ╪¿╪º╪▓╪▒╪│█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘ü╪▒╪º╪»╪º╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ auto meta = get\_reflection\_metadata(var\_ref); auto enhanced = create\_enhanced\_inspection(var\_ref, meta); return enhanced; } }; } // ┘à╪½╪º┘ä ╪│┘å╪º╪▒█î┘ê ╪»█î╪¿╪º┌» void debug\_example() { class $serializable $observable Person { std::string name; int age; }; Person p{"Alice", 30}; // ┌å╪º┘ä╪┤┘ç╪º█î ╪»█î╪¿╪º┌»: // 1. ╪¬┘å╪╕█î┘à breakpoint ╪»╪▒ ┘à╪¬╪» to\_json() ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç // 2. ╪¿╪º╪▓╪▒╪│█î meta-object ┘ç╪º ╪╖█î ╪»█î╪¿╪º┌» // 3. ╪»╪▒┌⌐ call stack ╪º╪▓ ╪╖╪▒█î┘é ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç // 4. ╪¿╪º╪▓╪▒╪│█î ┘à╪¬╪║█î╪▒ ╪¿╪º ╪º╪╣╪╢╪º█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç auto json = p.to\_json(); // ┘à╪¬╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç - ┘å█î╪º╪▓ ╪¿┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪»█î╪¿╪º┌» } ### 8.2.2 ┌å╪º┘ä╪┤┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î IDE IDE ┘ç╪º█î ┘à╪»╪▒┘å ╪¿╪º█î╪» ╪¿╪▒╪º█î ╪º╪▒╪º╪ª┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à┘å╪º╪│╪¿ ╪º╪▓ ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘é┘ê█î╪¬ ╪┤┘ê┘å╪» [132]: cpp // ╪º┘ä╪▓╪º┘à╪º╪¬ █î┌⌐┘╛╪º╪▒┌å┌»█î IDE namespace ide\_support { class reflection\_language\_server { public: // ╪¬┌⌐┘à█î┘ä ┌⌐╪» ╪¿╪▒╪º█î ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ std::vector<completion\_item> get\_reflection\_completions( const source\_position& cursor\_position, const compilation\_context& context) { std::vector<completion\_item> completions; // ╪º┌»╪▒ cursor ╪¿╪╣╪» ╪º╪▓ reflexpr( ╪º╪│╪¬ if (in\_reflexpr\_context(cursor\_position)) { auto available\_types = get\_available\_types(context); for (const auto& type : available\_types) { completions.emplace\_back(create\_type\_completion(type)); } } // ╪º┌»╪▒ cursor ╪¿╪╣╪» ╪º╪▓ ┘å┘é╪╖┘ç meta object ╪º╪│╪¬ if (in\_meta\_object\_member\_access(cursor\_position)) { auto meta\_obj\_type = infer\_meta\_object\_type(cursor\_position, context); auto available\_operations = get\_meta\_operations(meta\_obj\_type); for (const auto& op : available\_operations) { completions.emplace\_back(create\_operation\_completion(op)); } } return completions; } // go-to-definition ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç std::vector<definition\_location> find\_definitions( const source\_position& position, const compilation\_context& context) { std::vector<definition\_location> definitions; auto symbol = get\_symbol\_at\_position(position); if (is\_generated\_symbol(symbol)) { // ┘╛█î╪»╪º ┌⌐╪▒╪»┘å ┌⌐╪º╪▒╪¿╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ ┌⌐┘ç ╪º█î┘å ┘å┘à╪º╪» ╪▒╪º ╪¬┘ê┘ä█î╪» ┌⌐╪▒╪»┘ç auto generator = find\_generating\_metaclass(symbol); definitions.push\_back(generator.application\_site); // ┘ç┘à┌å┘å█î┘å ╪¬╪╣╪▒█î┘ü ┘å┘ê╪╣ ╪º╪╡┘ä█î ╪▒╪º ┘å╪┤╪º┘å ╪»┘ç█î╪» definitions.push\_back(generator.original\_definition); } else { // ╪¼╪│╪¬╪¼┘ê█î ╪¬╪╣╪▒█î┘ü ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» definitions = standard\_find\_definitions(position, context); } return definitions; } // ╪º╪╖┘ä╪º╪╣╪º╪¬ hover ╪¿╪▒╪º█î meta-object ┘ç╪º hover\_information get\_hover\_info( const source\_position& position, const compilation\_context& context) { auto symbol = get\_symbol\_at\_position(position); if (is\_meta\_object(symbol)) { return create\_meta\_object\_hover(symbol); } else if (is\_generated\_symbol(symbol)) { return create\_generated\_symbol\_hover(symbol); } else { return standard\_hover\_info(position, context); } } private: hover\_information create\_meta\_object\_hover(const symbol\_info& symbol) { hover\_information info; info.type = "Meta-object"; info.description = format\_meta\_object\_description(symbol); info.documentation = get\_meta\_object\_documentation(symbol); return info; } hover\_information create\_generated\_symbol\_hover(const symbol\_info& symbol) { hover\_information info; info.type = "Generated Symbol"; info.description = format\_generated\_symbol\_description(symbol); info.generation\_context = get\_generation\_context(symbol); return info; } }; // ┘å╪¡┘ê highlighting ╪¿╪▒╪º█î ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿ class reflection\_syntax\_highlighter { public: syntax\_highlighting\_result highlight\_reflection\_code( const source\_text& text) { syntax\_highlighting\_result result; // highlighting ╪╣┘à┘ä┌»╪▒┘ç╪º█î reflexpr highlight\_reflexpr\_operators(text, result); // highlighting ╪╣┘à┘ä█î╪º╪¬ meta-object highlight\_meta\_operations(text, result); // highlighting ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ highlight\_metaclass\_applications(text, result); // highlighting ┘å╪┤╪º┘å┌»╪▒┘ç╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç highlight\_generated\_code\_markers(text, result); return result; } }; } ## 8.3 ┌⌐█î┘ü█î╪¬ ┘╛█î╪º┘à ╪«╪╖╪º ### 8.3.1 ╪¬┌⌐╪½█î╪▒ ╪«╪╖╪º█î ┘é╪º┘ä╪¿ ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪«█î ┘╛█î┌å█î╪»┌»█î┘ç╪º█î ┘é╪º┘ä╪¿ ╪▒╪º ┌⌐╪º┘ç╪┤ ┘à█î╪»┘ç╪»╪î ┘à█î╪¬┘ê╪º┘å╪» ╪¿┘ç ╪»╪│╪¬┘ç┘ç╪º█î ╪¼╪»█î╪»█î ╪º╪▓ ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º█î ┘╛█î┌å█î╪»┘ç ┘å█î╪▓ ┘à┘å╪¼╪▒ ╪┤┘ê╪» [133]: cpp // ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪«╪╖╪º█î ┘╛█î┌å█î╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ template<typename T> void problematic\_reflection\_usage() { constexpr auto meta = std::meta::reflexpr(T); // ╪«╪╖╪º 1: ╪╣┘à┘ä█î╪º╪¬ ┘å╪º┘à╪╣╪¬╪¿╪▒ meta-object constexpr auto invalid = std::meta::get\_name\_v<meta>; // T ┘à┘à┌⌐┘å ╪º╪│╪¬ ┘å╪º┘à╪»╪º╪▒ ┘å╪¿╪º╪┤╪» // ╪«╪╖╪º 2: ╪¬╪╣╪º┘à┘ä╪º╪¬ ┘╛█î┌å█î╪»┘ç template-╪¿╪º╪▓╪¬╪º╪¿ constexpr auto members = std::meta::data\_members\_of(meta); std::meta::template\_for<members>([](auto member) { // ╪«╪╖╪º┘ç╪º█î ┘é╪º┘ä╪¿ ╪¬┘ê╪»╪▒╪¬┘ê ╪»╪▒ ╪¡┘ä┘é┘ç┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ constexpr auto member\_type = std::meta::get\_type\_t<member>; if constexpr (requires { typename some\_complex\_trait<member\_type>::type; }) { // ╪¬╪╣╪º┘à┘ä╪º╪¬ ┘╛█î┌å█î╪»┘ç SFINAE ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ some\_complex\_operation<member\_type>(); } }); // ╪«╪╖╪º 3: ┘å┘é╪╢ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ static\_assert(satisfies\_metaclass\_constraints<T>(), "Type does not satisfy metaclass requirements"); } // ┘à╪½╪º┘ä ╪¿┘ç╪¿┘ê╪» ┘╛█î╪º┘à ╪«╪╖╪º ┘à┘ê╪▒╪» ┘å█î╪º╪▓: /\* ╪«╪╖╪º█î ╪│┘å╪¬█î: error: no matching function for call to 'some\_complex\_operation<anonymous>' note: candidate template ignored: substitution failure [with T = (lambda at file.cpp:15:42)] note: in instantiation of function template specialization 'problematic\_reflection\_usage<MyClass>' requested here ╪«╪╖╪º█î ╪¿┘ç╪¿┘ê╪» █î╪º┘ü╪¬┘ç ┘à╪╖┘ä┘ê╪¿: error: reflection operation failed in metaclass application note: while processing member 'invalid\_member' of type 'MyClass' note: member type 'std::unique\_ptr<NonSerializable>' does not satisfy serialization constraints note: consider adding custom serialization for 'NonSerializable' or marking member as transient \*/ ### 8.3.2 ╪▓┘à█î┘å┘ç ╪«╪╖╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪«╪╖╪º┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘å█î╪º╪▓ ╪¿┘ç ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º█î ╪¬╪«╪╡╪╡█î ╪¿╪▒╪º█î ╪º╪▒╪º╪ª┘ç ╪¿╪º╪▓╪«┘ê╪▒╪» ┘à╪╣┘å╪º╪»╪º╪▒ ╪»╪º╪▒┘å╪» [134]: cpp // ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪▒╪º█î ╪╣┘à┘ä█î╪º╪¬ ┘à╪¬╪º┌⌐┘ä╪º╪│ namespace error\_reporting { class metaclass\_error\_context { struct error\_context\_frame { source\_location metaclass\_application; std::string metaclass\_name; source\_location target\_type\_definition; std::string current\_operation; std::optional<member\_info> current\_member; }; std::vector<error\_context\_frame> context\_stack\_; public: void push\_context(const std::string& metaclass\_name, const source\_location& application\_site, const source\_location& target\_definition) { context\_stack\_.emplace\_back(error\_context\_frame{ .metaclass\_application = application\_site, .metaclass\_name = metaclass\_name, .target\_type\_definition = target\_definition, .current\_operation = "", .current\_member = std::nullopt }); } void set\_current\_operation(const std::string& operation) { if (!context\_stack\_.empty()) { context\_stack\_.back().current\_operation = operation; } } void set\_current\_member(const member\_info& member) { if (!context\_stack\_.empty()) { context\_stack\_.back().current\_member = member; } } std::string format\_error\_message(const std::string& base\_error) const { std::ostringstream msg; msg << base\_error << "\n"; if (!context\_stack\_.empty()) { const auto& top = context\_stack\_.back(); msg << "note: in metaclass '" << top.metaclass\_name << "' " << "applied at " << format\_location(top.metaclass\_application) << "\n"; if (!top.current\_operation.empty()) { msg << "note: while " << top.current\_operation << "\n"; } if (top.current\_member) { msg << "note: processing member '" << top.current\_member->name << "' of type '" << top.current\_member->type\_name << "'\n"; } msg << "note: target type defined at " << format\_location(top.target\_type\_definition) << "\n"; } return msg.str(); } }; // ╪▓┘à█î┘å┘ç ╪«╪╖╪º█î ╪│╪▒╪º╪│╪▒█î ╪¿╪▒╪º█î ╪╣┘à┘ä█î╪º╪¬ ┘à╪¬╪º┌⌐┘ä╪º╪│ thread\_local metaclass\_error\_context current\_metaclass\_context; // ┘à╪»█î╪▒█î╪¬ ╪▓┘à█î┘å┘ç RAII class metaclass\_operation\_scope { bool context\_pushed\_; public: metaclass\_operation\_scope(const std::string& metaclass\_name, const source\_location& application\_site, const source\_location& target\_definition) : context\_pushed\_(true) { current\_metaclass\_context.push\_context(metaclass\_name, application\_site, target\_definition); } ~metaclass\_operation\_scope() { if (context\_pushed\_) { current\_metaclass\_context.pop\_context(); } } void set\_operation(const std::string& operation) { current\_metaclass\_context.set\_current\_operation(operation); } void set\_member(const member\_info& member) { current\_metaclass\_context.set\_current\_member(member); } }; } ## 8.4 ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î ┘ê ┘à┘ê╪º┘å╪╣ ┘╛╪░█î╪▒╪┤ ### 8.4.1 ┘╛█î┌å█î╪»┌»█î ┘à┘ü┘ç┘ê┘à█î ┘à╪╣╪▒┘ü█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘╛█î┌å█î╪»┌»█î ┘à┘ü┘ç┘ê┘à█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿┘ç C++ ╪º╪╢╪º┘ü┘ç ┘à█î┌⌐┘å╪» [135]: cpp // ┘ä╪º█î┘ç┘ç╪º█î ┘╛█î┌å█î╪»┌»█î ╪»╪▒ ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ namespace complexity\_analysis { // ┘ä╪º█î┘ç 1: ┘à┘ü╪º┘ç█î┘à ┘╛╪º█î┘ç ╪¿╪º╪▓╪¬╪º╪¿ void basic\_reflection\_concepts() { // ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪¿╪º█î╪» ╪¿┘ü┘ç┘à╪»: // - Meta-object ┘ç╪º ╪»╪▒ ┘à┘é╪º╪¿┘ä ╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î ┘à╪╣┘à┘ê┘ä█î // - ╪¬┘à╪º█î╪▓╪º╪¬ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ ┘à┘é╪º╪¿┘ä ╪▓┘à╪º┘å ╪º╪¼╪▒╪º // - ╪▓┘à█î┘å┘ç┘ç╪º█î ╪º╪▒╪▓█î╪º╪¿█î constexpr struct Example { int member; }; constexpr auto meta = std::meta::reflexpr(Example); // ╪º█î╪¼╪º╪» Meta-object constexpr auto members = std::meta::data\_members\_of(meta); // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î Meta-object constexpr auto size = std::meta::get\_size\_v<members>; // ╪º╪▒╪▓█î╪º╪¿█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä } // ┘ä╪º█î┘ç 2: ╪¬╪╣╪º┘à┘ä╪º╪¬ template-╪¿╪º╪▓╪¬╪º╪¿ template<typename T> void template\_reflection\_interaction() { // ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪¿╪º█î╪» ╪¿┘ü┘ç┘à╪»: // - ╪▓┘à╪º┘å╪¿┘å╪»█î instantiation ┘é╪º┘ä╪¿ // - ╪»╪│╪¬╪▒╪│█î Meta-object ╪╖█î instantiation // - SFINAE ╪¿╪º ┘à╪¡┘à┘ê┘ä┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ constexpr auto meta = std::meta::reflexpr(T); if constexpr (std::meta::is\_class\_v<meta>) { // ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤╪▒╪╖█î ╪¿╪▒ ╪º╪│╪º╪│ ╪¿╪º╪▓╪¬╪º╪¿ process\_class\_type<T>(); } else { process\_non\_class\_type<T>(); } } // ┘ä╪º█î┘ç 3: ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ constexpr void advanced\_metaclass(std::meta::info target) { // ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪¿╪º█î╪» ╪¿┘ü┘ç┘à╪»: // - ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» // - ┘é┘ê╪º┘å█î┘å ╪¬╪▒┌⌐█î╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ // - ╪º╪▒╪¬╪¿╪º╪╖ ╪¿█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º // - ┘à╪»█î╪▒█î╪¬ ┘ê╪º╪¿╪│╪¬┌»█î validate\_metaclass\_preconditions(target); generate\_base\_functionality(target); integrate\_with\_other\_metaclasses(target); emit\_final\_code(target); } // ┘ä╪º█î┘ç 4: █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ template<Reflectable T> auto create\_async\_processor() -> std::generator<processed\_result<T>> { // ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪¿╪º█î╪» ╪¿┘ü┘ç┘à╪»: // - Concept ┘ç╪º + ╪¿╪º╪▓╪¬╪º╪¿ // - Coroutine ┘ç╪º + ╪¿╪º╪▓╪¬╪º╪¿ // - Range ┘ç╪º + ╪¿╪º╪▓╪¬╪º╪¿ // - Module ┘ç╪º + ╪¿╪º╪▓╪¬╪º╪¿ constexpr auto meta = std::meta::reflexpr(T); for (auto item : get\_input\_range<T>()) { auto processed = co\_await process\_with\_reflection(item, meta); co\_yield processed; } } } ### 8.4.2 ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘à┘ç╪º╪¼╪▒╪¬ ╪│╪º╪▓┘à╪º┘å┘ç╪º ╪¿╪º ┌å╪º┘ä╪┤┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪»╪▒ ┘à┘ç╪º╪¼╪▒╪¬ ┌⌐╪»╪¿█î╪│┘ç╪º█î ┘à┘ê╪¼┘ê╪» ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘ê╪º╪¼┘ç ┘ç╪│╪¬┘å╪» [136]: cpp // ╪¬╪¡┘ä█î┘ä ┘╛█î┌å█î╪»┌»█î ┘à┘ç╪º╪¼╪▒╪¬ namespace migration { // ┘à╪▒╪¡┘ä┘ç 1: ╪º╪▒╪▓█î╪º╪¿█î ┌⌐╪»╪¿█î╪│ ┘à┘ê╪¼┘ê╪» class codebase\_analysis { public: struct migration\_assessment { size\_t total\_types; size\_t serializable\_types; size\_t complex\_template\_hierarchies; size\_t manual\_code\_generation\_usage; std::vector<potential\_reflection\_opportunity> opportunities; std::vector<migration\_blocker> blockers; }; migration\_assessment analyze\_codebase(const codebase& code) { migration\_assessment result; // ╪┤┘å╪º╪│╪º█î█î ╪º┘å┘ê╪º╪╣█î ┌⌐┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç╪▒┘ç┘à┘å╪» ╪┤┘ê┘å╪» result.opportunities = find\_reflection\_opportunities(code); // ╪┤┘å╪º╪│╪º█î█î ┘à┘ê╪º┘å╪╣ ┘à┘ç╪º╪¼╪▒╪¬ result.blockers = find\_migration\_blockers(code); return result; } private: std::vector<potential\_reflection\_opportunity> find\_reflection\_opportunities( const codebase& code) { std::vector<potential\_reflection\_opportunity> opportunities; // ╪¼╪│╪¬╪¼┘ê ╪¿╪▒╪º█î ┌⌐╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¬┌⌐╪▒╪º╪▒█î auto serialization\_patterns = find\_serialization\_patterns(code); for (const auto& pattern : serialization\_patterns) { opportunities.emplace\_back(create\_serialization\_opportunity(pattern)); } // ╪¼╪│╪¬╪¼┘ê ╪¿╪▒╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪»╪│╪¬█î property auto property\_patterns = find\_property\_patterns(code); for (const auto& pattern : property\_patterns) { opportunities.emplace\_back(create\_property\_opportunity(pattern)); } return opportunities; } std::vector<migration\_blocker> find\_migration\_blockers(const codebase& code) { std::vector<migration\_blocker> blockers; // ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘å╪│╪«┘ç ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ if (!supports\_reflection(get\_compiler\_version())) { blockers.emplace\_back(migration\_blocker{ .type = blocker\_type::compiler\_support, .description = "Compiler does not support C++23 reflection" }); } // ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ┌⌐┘ç ┘à┘ç╪º╪¼╪▒╪¬ ╪ó┘å ╪»╪┤┘ê╪º╪▒ ╪º╪│╪¬ auto complex\_templates = find\_complex\_template\_usage(code); for (const auto& usage : complex\_templates) { if (is\_migration\_difficult(usage)) { blockers.emplace\_back(create\_template\_migration\_blocker(usage)); } } return blockers; } }; // ┘à╪▒╪¡┘ä┘ç 2: ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ┘à┘ç╪º╪¼╪▒╪¬ ╪º┘ü╪▓╪º█î╪┤█î class incremental\_migration\_planner { public: struct migration\_plan { std::vector<migration\_phase> phases; timeline estimated\_timeline; resource\_requirements resources; risk\_assessment risks; }; migration\_plan create\_migration\_plan(const migration\_assessment& assessment) { migration\_plan plan; // ┘à╪▒╪¡┘ä┘ç 1: ┘ü╪▒╪╡╪¬┘ç╪º█î ┌⌐┘à ╪«╪╖╪▒╪î ╪º╪▒╪▓╪┤ ╪¿╪º┘ä╪º auto phase1 = create\_low\_risk\_phase(assessment.opportunities); plan.phases.push\_back(phase1); // ┘à╪▒╪¡┘ä┘ç 2: ┘à┘ç╪º╪¼╪▒╪¬┘ç╪º█î ┘╛█î┌å█î╪»┌»█î ┘à╪¬┘ê╪│╪╖ auto phase2 = create\_medium\_complexity\_phase(assessment.opportunities); plan.phases.push\_back(phase2); // ┘à╪▒╪¡┘ä┘ç 3: ┘à┘ç╪º╪¼╪▒╪¬┘ç╪º█î ┘╛█î┌å█î╪»┌»█î ╪¿╪º┘ä╪º auto phase3 = create\_high\_complexity\_phase(assessment.opportunities); plan.phases.push\_back(phase3); return plan; } private: migration\_phase create\_low\_risk\_phase( const std::vector<potential\_reflection\_opportunity>& opportunities) { migration\_phase phase; phase.name = "Low-Risk Reflection Adoption"; phase.description = "Migrate simple serialization and property patterns"; // ╪¬┘à╪▒┌⌐╪▓ ╪¿╪▒ ╪º┘å┘ê╪º╪╣ ┘à╪│╪¬┘é┘ä ╪¿╪º ┘ê╪º╪¿╪│╪¬┌»█î┘ç╪º█î ╪¡╪»╪º┘é┘ä for (const auto& opp : opportunities) { if (opp.risk\_level == risk\_level::low && opp.value\_impact == impact\_level::high) { phase.tasks.push\_back(create\_migration\_task(opp)); } } return phase; } }; } ## 8.5 ┌å╪º┘ä╪┤┘ç╪º█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ### 8.5.1 ┘å┌»╪▒╪º┘å█î┘ç╪º█î ╪½╪¿╪º╪¬ ABI ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┌å╪º┘ä╪┤┘ç╪º█î█î ╪¿╪▒╪º█î ╪½╪¿╪º╪¬ ╪▒╪º╪¿╪╖ ╪¿╪º█î┘å╪▒█î ┌⌐╪º╪▒╪¿╪▒╪» (ABI) ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å┘å╪» [137]: cpp // ┘à┘ä╪º╪¡╪╕╪º╪¬ ╪½╪¿╪º╪¬ ABI namespace abi\_stability { // ┘à╪┤┌⌐┘ä: ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪▒┘ê█î ABI ╪¬╪ú╪½█î╪▒ ┘à█î┌»╪░╪º╪▒╪» class $serializable Version1 { int id; std::string name; // ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç: to\_json(), from\_json(), operator==, etc. }; class $serializable Version2 { int id; std::string name; std::string email; // ┘ü█î┘ä╪» ╪º╪╢╪º┘ü┘ç ╪┤╪»┘ç // ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç: to\_json(), from\_json(), operator==, etc. }; // ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪┤┌⌐╪│╪¬ ABI: // 1. ╪º┘à╪╢╪º┘ç╪º█î ┘à╪¬╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬╪║█î█î╪▒ ┘à█î┌⌐┘å┘å╪» // 2. ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à╪¬╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬╪║█î█î╪▒ ┘à█î┌⌐┘å┘å╪» // 3. ╪¼╪»╪º┘ê┘ä ┘à╪¼╪º╪▓█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬╪║█î█î╪▒ ┘à█î┌⌐┘å┘å╪» // 4. ┌å█î╪»┘à╪º┘å ╪»╪º╪»┘ç ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¬╪║█î█î╪▒ ┘à█î┌⌐┘å╪» struct abi\_stability\_analysis { enum class compatibility\_level { source\_compatible, // ┌⌐╪» ┘à┘å╪¿╪╣ ┌⌐╪º┘à┘╛╪º█î┘ä ┘à█î╪┤┘ê╪» binary\_compatible, // ╪¿╪º█î┘å╪▒█î┘ç╪º█î ┘à┘ê╪¼┘ê╪» ┌⌐╪º╪▒ ┘à█î┌⌐┘å┘å╪» runtime\_compatible // ╪▒┘ü╪¬╪º╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¡┘ü╪╕ ┘à█î╪┤┘ê╪» }; static compatibility\_level analyze\_metaclass\_change( const metaclass\_definition& old\_def, const metaclass\_definition& new\_def) { // ╪¬╪¡┘ä█î┘ä ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç auto old\_generated = simulate\_code\_generation(old\_def); auto new\_generated = simulate\_code\_generation(new\_def); if (old\_generated.signatures != new\_generated.signatures) { return compatibility\_level::source\_compatible; } if (old\_generated.implementations != new\_generated.implementations) { return compatibility\_level::binary\_compatible; } return compatibility\_level::runtime\_compatible; } }; // ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ┘å╪│╪«┘ç╪¿┘å╪»█î ╪¿╪▒╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º class metaclass\_versioning { public: // ┘å╪│╪«┘ç╪¿┘å╪»█î ╪╡╪▒█î╪¡ ╪¿╪▒╪º█î ╪½╪¿╪º╪¬ ABI constexpr void serializable\_v1(std::meta::info target) { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘å╪│╪«┘ç 1 - ABI ┘╛╪º█î╪»╪º╪▒ generate\_json\_methods\_v1(target); } constexpr void serializable\_v2(std::meta::info target) { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘å╪│╪«┘ç 2 - ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ABI ╪┤┌⌐┘å generate\_json\_methods\_v2(target); generate\_validation\_methods(target); } // ┘╛█î╪┤┘ü╪▒╪╢ ╪¿┘ç ╪ó╪«╪▒█î┘å ┘å╪│╪«┘ç ┘╛╪º█î╪»╪º╪▒ constexpr void serializable(std::meta::info target) { serializable\_v1(target); // ┘╛█î╪┤┘ü╪▒╪╢ ┘à╪¡╪º┘ü╪╕┘ç┌⌐╪º╪▒╪º┘å┘ç } }; } ### 8.5.2 ╪│╪º╪▓┌»╪º╪▒█î ╪¿█î┘å ┘ü╪▒┘ê╪┤┘å╪»┌»╪º┘å ╪º╪╖┘à█î┘å╪º┘å ╪º╪▓ ╪▒┘ü╪¬╪º╪▒ ╪½╪º╪¿╪¬ ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à╪«╪¬┘ä┘ü ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┌å╪º┘ä╪┤┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [138]: cpp // ┌å╪º┘ä╪┤┘ç╪º█î ╪│╪º╪▓┌»╪º╪▒█î ╪¿█î┘å ┘ü╪▒┘ê╪┤┘å╪»┌»╪º┘å namespace vendor\_compatibility { // ┌⌐╪º┘à┘╛╪º█î┘ä╪▒┘ç╪º█î ┘à╪«╪¬┘ä┘ü ┘à┘à┌⌐┘å ╪º╪│╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ┘à╪¬┘ü╪º┘ê╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐┘å┘å╪» struct compiler\_specific\_behavior { enum class vendor { gcc, clang, msvc, icc }; // ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ┘å┘à╪º█î╪┤ Meta-object static bool are\_meta\_objects\_equivalent( const std::meta::info& obj1, const std::meta::info& obj2, vendor v1, vendor v2) { if (v1 == v2) { return obj1 == obj2; // ┘à┘é╪º█î╪│┘ç ┘ü╪▒┘ê╪┤┘å╪»┘ç █î┌⌐╪│╪º┘å } // ┘à┘é╪º█î╪│┘ç ╪¿█î┘å ┘ü╪▒┘ê╪┤┘å╪»┌»╪º┘å ┘å█î╪º╪▓ ╪¿┘ç ┘å╪▒┘à╪º┘ä╪│╪º╪▓█î ╪»╪º╪▒╪» return normalize\_meta\_object(obj1, v1) == normalize\_meta\_object(obj2, v2); } // ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» static std::string normalize\_generated\_code( const std::string& generated\_code, vendor source\_vendor) { // ┘å╪▒┘à╪º┘ä╪│╪º╪▓█î ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ╪«╪º╪╡ ┌⌐╪º┘à┘╛╪º█î┘ä╪▒: // - ╪¬┘å┘ê╪╣╪º╪¬ name mangling // - ╪¬┘ü╪º┘ê╪¬┘ç╪º█î instantiation ┘é╪º┘ä╪¿ // - ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ┘ü╪▒╪╢ ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î return apply\_normalization\_rules(generated\_code, source\_vendor); } }; // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘é╪º╪¿┘ä ╪¡┘à┘ä class portable\_metaclass { public: constexpr void portable\_serializable(std::meta::info target) { // ╪º╪¼╪¬┘å╪º╪¿ ╪º╪▓ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪«╪º╪╡ ┘ü╪▒┘ê╪┤┘å╪»┘ç if constexpr (supports\_advanced\_reflection()) { generate\_advanced\_serialization(target); } else { generate\_basic\_serialization(target); } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¬╪┤╪«█î╪╡ ┘ê█î┌ÿ┌»█î ╪¿┘ç ╪¼╪º█î ╪¬╪┤╪«█î╪╡ ┘ü╪▒┘ê╪┤┘å╪»┘ç if constexpr (has\_string\_literal\_templates()) { use\_string\_literal\_optimization(target); } } private: // ╪¬╪┤╪«█î╪╡ ┘ê█î┌ÿ┌»█î ╪¿╪▒╪º█î ┘é╪º╪¿┘ä█î╪¬ ╪¡┘à┘ä static consteval bool supports\_advanced\_reflection() { // ╪ó╪▓┘à╪º█î╪┤ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪»╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä return requires { std::meta::advanced\_query\_operation(); }; } static consteval bool has\_string\_literal\_templates() { // ╪ó╪▓┘à╪º█î╪┤ ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î ┘é╪º┘ä╪¿ literal ╪▒╪┤╪¬┘ç return requires { template\_with\_string\_literal<"test">(); }; } }; } ╪º█î┘å ┌å╪º┘ä╪┤┘ç╪º ┘╛█î┌å█î╪»┌»█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ┘╛╪░█î╪▒╪┤ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪»┘å█î╪º█î ┘ê╪º┘é╪╣█î ╪▒╪º ╪¿╪▒╪¼╪│╪¬┘ç ┘à█î┌⌐┘å┘å╪». ╪¿╪º ┘ê╪¼┘ê╪» ╪º█î┘å ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º╪î ┘à╪▓╪º█î╪º ╪º╪║┘ä╪¿ ╪¿╪▒ ┘ç╪▓█î┘å┘ç┘ç╪º ╪║┘ä╪¿┘ç ┘à█î┌⌐┘å┘å╪»╪î ╪«╪º╪╡┘ç ╪¿╪▒╪º█î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î█î ┌⌐┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ┘é╪»╪▒╪¬ ┌⌐╪º┘à┘ä ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å┘å╪». ╪¿╪«╪┤ ╪¿╪╣╪»█î ╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç ╪¿╪▒╪º█î ╪▒┘ü╪╣ ╪º█î┘å ┌å╪º┘ä╪┤┘ç╪º ┘ê ┌»╪│╪¬╪▒╪┤ ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪». ΓÇö *[┘à╪▒╪º╪¼╪╣ 127-138 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒╪î ╪¬╪¡┘é█î┘é╪º╪¬ ╪▓█î╪▒╪│╪º╪«╪¬ ╪»█î╪¿╪º┌»╪î ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º╪î ╪¬╪¡┘ä█î┘ä ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î╪î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘à┘ç╪º╪¼╪▒╪¬ ┘ê ╪¬┘ä╪º╪┤┘ç╪º█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 9. ╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç ## 9.1 C++26 ┘ê ┘ü╪▒╪º╪¬╪▒ ╪º╪▓ ╪ó┘å: ┌»╪│╪¬╪▒╪┤ ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ### 9.1.1 ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç C++23 ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»╪î ╪╣┘ä╪º┘é┘ç ┘ü╪▓╪º█î┘å╪»┘ç╪º█î ╪¿╪▒╪º█î ┌»╪│╪¬╪▒╪┤ ╪º█î┘å ┘é╪º╪¿┘ä█î╪¬┘ç╪º ╪¿┘ç ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ê╪¼┘ê╪» ╪»╪º╪▒╪» [139]. ┌⌐┘à█î╪¬┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î C++ ╪¿┘ç ╪╖┘ê╪▒ ┘ü╪╣╪º┘ä ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î█î ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º ╪▒╪º ╪¿╪▒╪▒╪│█î ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘à┌⌐┘à┘ä ╪▓█î╪▒╪│╪º╪«╪¬ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘à┘ê╪¼┘ê╪» ╪¿╪º╪┤╪»: cpp // API ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º█î ┘╛█î╪┤┘å┘ç╪º╪»█î ╪¿╪▒╪º█î C++26 namespace std::meta::dynamic { // ╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê╪╣ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪º █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º╪▓╪¬╪º╪¿ class runtime\_type\_info { const std::meta::info static\_info\_; const std::type\_info& type\_info\_; public: // ┘╛┘ä ╪¿█î┘å ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ┘╛┘ê█î╪º constexpr runtime\_type\_info(std::meta::info static\_meta) : static\_info\_(static\_meta), type\_info\_(std::meta::get\_type\_info(static\_meta)) {} // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¬╪º╪»█î╪¬╪º█î ╪º╪│╪¬╪º╪¬█î┌⌐ std::vector<member\_descriptor> get\_members() const { // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪¬╪º╪»█î╪¬╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒╪º█î ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º constexpr auto static\_members = std::meta::data\_members\_of(static\_info\_); std::vector<member\_descriptor> result; std::meta::template\_for<static\_members>([&](auto member) { result.emplace\_back(create\_runtime\_descriptor(member)); }); return result; } // ╪»╪│╪¬╪▒╪│█î ╪╣╪╢┘ê ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪º ┘å╪º┘à std::optional<any\_value> get\_member\_value( const void\* object, std::string\_view member\_name) const { constexpr auto members = std::meta::data\_members\_of(static\_info\_); std::optional<any\_value> result; std::meta::template\_for<members>([&](auto member) { constexpr auto name = std::meta::get\_name\_v<member>; if (name == member\_name) { auto\* typed\_obj = static\_cast<const std::meta::get\_reflected\_type\_t<static\_info\_>\*>(object); result = get\_member\_value\_impl(typed\_obj, member); } }); return result; } // ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪¬╪» ╪▓┘à╪º┘å ╪º╪¼╪▒╪º std::optional<any\_value> invoke\_method( void\* object, std::string\_view method\_name, std::span<any\_value> arguments) const { constexpr auto methods = std::meta::member\_functions\_of(static\_info\_); std::optional<any\_value> result; std::meta::template\_for<methods>([&](auto method) { constexpr auto name = std::meta::get\_name\_v<method>; if (name == method\_name) { result = invoke\_method\_impl(object, method, arguments); } }); return result; } }; // ╪▒╪¼█î╪│╪¬╪▒█î ╪│╪▒╪º╪│╪▒█î ╪¿╪▒╪º█î ╪¼╪│╪¬╪¼┘ê█î ┘å┘ê╪╣ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º class type\_registry { std::unordered\_map<std::string, std::unique\_ptr<runtime\_type\_info>> registry\_; public: // ╪½╪¿╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ╪º┘å┘ê╪º╪╣ ╪¿╪º╪▓╪¬╪º╪¿█î template<typename T> void register\_type() { constexpr auto meta = std::meta::reflexpr(T); constexpr auto name = std::meta::get\_name\_v<meta>; registry\_[std::string(name)] = std::make\_unique<runtime\_type\_info>(meta); } // ╪¼╪│╪¬╪¼┘ê█î ┘å┘ê╪╣ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪º ┘å╪º┘à const runtime\_type\_info\* find\_type(std::string\_view type\_name) const { auto it = registry\_.find(std::string(type\_name)); return it != registry\_.end() ? it->second.get() : nullptr; } // ╪º█î╪¼╪º╪» ╪ó╪¿╪¼┌⌐╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪▓ ┘å╪º┘à ┘å┘ê╪╣ std::unique\_ptr<void, void(\*)(void\*)> create\_object(std::string\_view type\_name) const { auto\* type\_info = find\_type(type\_name); if (!type\_info) { return {nullptr, [](void\*){}}; } return type\_info->create\_default\_instance(); } }; // ┘å┘à┘ê┘å┘ç ╪▒╪¼█î╪│╪¬╪▒█î ┘å┘ê╪╣ ╪│╪▒╪º╪│╪▒█î inline type\_registry& get\_global\_registry() { static type\_registry registry; return registry; } } // ┘à╪½╪º┘ä ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º█î ┘╛█î╪┤┘å┘ç╪º╪»█î void dynamic\_reflection\_example() { using namespace std::meta::dynamic; // ╪½╪¿╪¬ ╪º┘å┘ê╪º╪╣ ╪¿╪▒╪º█î ╪¼╪│╪¬╪¼┘ê█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º get\_global\_registry().register\_type<Person>(); get\_global\_registry().register\_type<Company>(); // ╪º█î╪¼╪º╪» ┘ê ╪»╪│╪¬┌⌐╪º╪▒█î ╪ó╪¿╪¼┌⌐╪¬ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º auto obj = get\_global\_registry().create\_object("Person"); auto\* type\_info = get\_global\_registry().find\_type("Person"); if (type\_info && obj) { // ╪¬┘å╪╕█î┘à ┘à┘é╪º╪»█î╪▒ ╪╣╪╢┘ê ╪»╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º type\_info->set\_member\_value(obj.get(), "name", std::string("Alice")); type\_info->set\_member\_value(obj.get(), "age", 30); // ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪¬╪»┘ç╪º ╪»╪▒ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º auto result = type\_info->invoke\_method(obj.get(), "to\_string", {}); if (result) { std::cout << "Object string representation: " << std::any\_cast<std::string>(\*result) << std::endl; } } } ### 9.1.2 ╪¬╪▒┌⌐█î╪¿ ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î ╪ó█î┘å╪»┘ç C++ ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î ╪¬╪▒┌⌐█î╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘╛█î┌å█î╪»┘ç╪¬╪▒█î ╪▒╪º ┘à╪╣╪▒┘ü█î ╪«┘ê╪º┘ç┘å╪» ┌⌐╪▒╪» [140]: cpp // ╪¬╪▒┌⌐█î╪¿ ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î C++26 namespace future\_metaclasses { // ┘ê╪▒╪º╪½╪¬ ┘ê ╪¬╪▒┌⌐█î╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ template<typename Base> constexpr void derived\_metaclass(std::meta::info target) requires IsMetaclass<Base> { // ╪º╪¿╪¬╪»╪º ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘╛╪º█î┘ç ╪▒╪º ╪º╪╣┘à╪º┘ä ┌⌐┘å apply\_metaclass<Base>(target); // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪┤╪¬┘é ╪┤╪»┘ç add\_derived\_functionality(target); // ╪▒┘ü╪¬╪º╪▒┘ç╪º█î ┘╛╪º█î┘ç ╪«╪º╪╡ ╪▒╪º ╪¿╪º╪▓┘å┘ê█î╪│█î ┌⌐┘å override\_base\_methods(target); } // ┌⌐╪º╪▒╪¿╪▒╪» ┌å┘å╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º ╪¡┘ä ╪¬╪╣╪º╪▒╪╢ template<typename... Metaclasses> constexpr void combined\_metaclass(std::meta::info target) { // ╪º╪╣┘à╪º┘ä ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪¬╪▒╪¬█î╪¿ ╪¿╪º ╪¬╪┤╪«█î╪╡ ╪¬╪╣╪º╪▒╪╢ apply\_metaclasses\_with\_resolution<Metaclasses...>(target); } // ┌⌐╪º╪▒╪¿╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪┤╪▒╪╖█î template<typename Condition, typename ThenMetaclass, typename ElseMetaclass = void> constexpr void conditional\_metaclass(std::meta::info target) { if constexpr (Condition::evaluate(target)) { apply\_metaclass<ThenMetaclass>(target); } else if constexpr (!std::is\_void\_v<ElseMetaclass>) { apply\_metaclass<ElseMetaclass>(target); } } // ╪¼┘å╪¿┘ç┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î ┘å┌»╪▒╪º┘å█î┘ç╪º█î ┘à╪¬┘é╪º╪¿┘ä namespace aspects { constexpr void logging\_aspect(std::meta::info target) { // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘ä╪º┌» ╪¿┘ç ┘ç┘à┘ç ┘à╪¬╪»┘ç╪º█î ╪╣┘à┘ê┘à█î auto methods = std::meta::member\_functions\_of(target); std::meta::template\_for<methods>([](auto method) { if (std::meta::is\_public\_v<method>) { wrap\_method\_with\_logging(method); } }); } constexpr void performance\_aspect(std::meta::info target) { // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘å╪╕╪º╪▒╪¬ ┌⌐╪º╪▒╪º█î█î ╪¿┘ç ┘à╪¬╪»┘ç╪º auto methods = std::meta::member\_functions\_of(target); std::meta::template\_for<methods>([](auto method) { if (should\_monitor\_performance(method)) { wrap\_method\_with\_timing(method); } }); } constexpr void security\_aspect(std::meta::info target) { // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ╪¿╪▒╪▒╪│█î┘ç╪º█î ╪º┘à┘å█î╪¬█î ╪¿┘ç ┘à╪¬╪»┘ç╪º█î ╪¡╪│╪º╪│ auto methods = std::meta::member\_functions\_of(target); std::meta::template\_for<methods>([](auto method) { if (has\_security\_annotation(method)) { wrap\_method\_with\_security\_check(method); } }); } } // ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪º ╪¬╪▒┌⌐█î╪¿ ╪¼┘å╪¿┘ç constexpr void enterprise\_entity(std::meta::info target) { // ╪º╪╣┘à╪º┘ä ╪╣┘à┘ä┌⌐╪▒╪» ╪º╪╡┘ä█î entity apply\_metaclass<serializable>(target); apply\_metaclass<observable>(target); apply\_metaclass<validatable>(target); // ╪º╪╣┘à╪º┘ä ╪¼┘å╪¿┘ç┘ç╪º█î ┘à╪¬┘é╪º╪¿┘ä apply\_aspect<aspects::logging\_aspect>(target); apply\_aspect<aspects::performance\_aspect>(target); apply\_aspect<aspects::security\_aspect>(target); // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪«╪º╪╡ ╪│╪º╪▓┘à╪º┘å█î generate\_audit\_trail\_support(target); generate\_versioning\_support(target); generate\_caching\_support(target); } } // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¬╪▒┌⌐█î╪¿ ┘╛█î╪┤╪▒┘ü╪¬┘ç class $enterprise\_entity Person { std::string name; int age; std::string ssn [[security::sensitive]]; void update\_profile(const std::string& new\_name) [[performance::monitor]] { name = new\_name; } }; ### 9.1.3 ╪¬┘é┘ê█î╪¬ █î┌⌐┘╛╪º╪▒┌å┌»█î ┘à╪º┌ÿ┘ê┘ä ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î ╪ó█î┘å╪»┘ç ╪º╪¡╪¬┘à╪º┘ä╪º┘ï █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿┘ç╪¬╪▒█î ╪¿█î┘å ╪¿╪º╪▓╪¬╪º╪¿╪î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘ê ╪│█î╪│╪¬┘à ┘à╪º┌ÿ┘ê┘ä┘ç╪º ╪º╪▒╪º╪ª┘ç ╪«┘ê╪º┘ç┘å╪» ╪»╪º╪» [141]: cpp // █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ┘à╪º┌ÿ┘ê┘ä-╪¿╪º╪▓╪¬╪º╪¿ export module person\_model; import std.meta; import std.reflection.serialization; import std.reflection.orm; // ╪º╪╣┘ä╪º┘å ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪ó┌»╪º┘ç ╪º╪▓ ┘à╪º┌ÿ┘ê┘ä export namespace model\_metaclasses { // ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ┘à╪º┌ÿ┘ê┘ä┘ç╪º ╪╡╪º╪»╪▒ ╪┤┘ê┘å╪» export constexpr void domain\_entity(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪ó┌»╪º┘ç ╪º╪▓ ┘à╪º┌ÿ┘ê┘ä generate\_in\_module\_context(target, get\_current\_module()); // ╪▒╪»█î╪º╪¿█î ┘ê╪º╪¿╪│╪¬┌»█î ╪¿█î┘å ┘à╪º┌ÿ┘ê┘ä█î register\_cross\_module\_dependencies(target); } // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«╪º╪╡ ┘à╪º┌ÿ┘ê┘ä export constexpr void json\_serializable(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ┌⌐╪»█î ┌⌐┘ç ┘à╪▒╪▓┘ç╪º█î ┘à╪º┌ÿ┘ê┘ä ╪▒╪º ╪▒╪╣╪º█î╪¬ ┌⌐┘å╪» generate\_module\_aware\_serialization(target); // ╪╡╪º╪»╪▒╪º╪¬ ┘à┘å╪º╪│╪¿ ╪¬┘ê╪º╪¿╪╣ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î auto serialization\_functions = generate\_serialization\_code(target); export\_functions\_from\_module(serialization\_functions); } } // ╪▒╪¼█î╪│╪¬╪▒█î ┘å┘ê╪╣ ┘à╪¡╪»┘ê╪» ╪¿┘ç ┘à╪º┌ÿ┘ê┘ä export namespace module\_registry { // ╪▒╪¼█î╪│╪¬╪▒█î ┘à╪¡╪»┘ê╪» ╪¿┘ç ╪º█î┘å ┘à╪º┌ÿ┘ê┘ä class module\_type\_registry { static inline std::vector<std::meta::info> registered\_types\_; public: template<typename T> static void register\_type() { constexpr auto meta = std::meta::reflexpr(T); registered\_types\_.push\_back(meta); } static auto get\_registered\_types() { return registered\_types\_; } }; // ╪½╪¿╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ╪º┘å┘ê╪º╪╣ ╪»╪▒ ╪º█î┘å ┘à╪º┌ÿ┘ê┘ä template<typename T> void auto\_register() { module\_type\_registry::register\_type<T>(); } } // ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿█î┘å ┘à╪º┌ÿ┘ê┘ä█î export namespace cross\_module { // ╪¼╪│╪¬╪¼┘ê█î ╪º┘å┘ê╪º╪╣ ╪»╪▒ ┘à╪▒╪▓┘ç╪º█î ┘à╪º┌ÿ┘ê┘ä template<typename Predicate> auto find\_types\_across\_modules(Predicate pred) { std::vector<std::meta::info> results; // ╪¼╪│╪¬╪¼┘ê█î ┘à╪º┌ÿ┘ê┘ä ┘ü╪╣┘ä█î auto local\_types = module\_registry::module\_type\_registry::get\_registered\_types(); for (auto type : local\_types) { if (pred(type)) { results.push\_back(type); } } // ╪¼╪│╪¬╪¼┘ê█î ┘à╪º┌ÿ┘ê┘ä┘ç╪º█î ┘ê╪º╪▒╪»╪º╪¬█î (┘ê█î┌ÿ┌»█î ╪ó█î┘å╪»┘ç) auto imported\_types = get\_imported\_module\_types(); for (auto type : imported\_types) { if (pred(type)) { results.push\_back(type); } } return results; } } ## 9.2 █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º█î ┘å┘ê╪╕┘ç┘ê╪▒ ### 9.2.1 █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¬╪▒┌⌐█î╪¿ ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ┘ü╪▒╪╡╪¬┘ç╪º█î ┘ç█î╪¼╪º┘å╪º┘å┌»█î╪▓█î ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪» ┘ê ╪¬┘ê┘ä█î╪» ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [142]: cpp // ╪¬┘ê┘ä█î╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º ML namespace ml\_enhanced { // ┘à╪»┘ä █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ╪¿╪▒╪º█î ╪¬╪┤╪«█î╪╡ ╪º┘ä┌»┘ê█î ┌⌐╪» class code\_pattern\_analyzer { // ┘à╪»┘ä ML ╪ó┘à┘ê╪▓╪┤ ╪»█î╪»┘ç ╪▒┘ê█î ┌⌐╪»╪¿█î╪│┘ç╪º ╪¿╪▒╪º█î ╪¬╪┤╪«█î╪╡ ╪º┘ä┌»┘ê┘ç╪º ml\_model pattern\_recognition\_model\_; public: // ╪¬╪¡┘ä█î┘ä ╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┘å┘ê╪╣ ╪¿╪▒╪º█î ┘╛█î╪┤┘å┘ç╪º╪» ╪╖╪▒╪º╪¡█î ╪¿┘ç█î┘å┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ metaclass\_suggestions analyze\_type\_usage(std::meta::info type) { // ╪º╪│╪¬╪«╪▒╪º╪¼ ┘ê█î┌ÿ┌»█î┘ç╪º ╪º╪▓ ┘à╪¬╪º╪»█î╪¬╪º█î ┘å┘ê╪╣ auto features = extract\_type\_features(type); // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┘à╪»┘ä ML ╪¿╪▒╪º█î ┘╛█î╪┤╪¿█î┘å█î ┘╛█î┌⌐╪▒╪¿┘å╪»█î ╪¿┘ç█î┘å┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ auto predictions = pattern\_recognition\_model\_.predict(features); return convert\_predictions\_to\_suggestions(predictions); } // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç optimized\_code\_generation optimize\_generated\_code( const generated\_code& base\_code, const usage\_statistics& stats) { // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪» ╪▒╪º┘ç┘å┘à╪º█î█î ╪┤╪»┘ç ╪¿╪º ML auto optimization\_strategy = pattern\_recognition\_model\_.suggest\_optimizations( base\_code, stats); return apply\_optimizations(base\_code, optimization\_strategy); } }; // ╪¬┘ê╪│╪╣┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º ┌⌐┘à┌⌐ AI constexpr void ai\_optimized\_serializable(std::meta::info target) { // ╪¬╪¡┘ä█î┘ä ┘à╪┤╪«╪╡╪º╪¬ ┘å┘ê╪╣ auto characteristics = analyze\_type\_characteristics(target); // ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ AI ╪¿╪▒╪º█î ╪¬╪╣█î█î┘å ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ╪¿┘ç█î┘å┘ç ╪│╪▒█î╪º┘ä╪│╪º╪▓█î auto strategy = ai\_suggest\_serialization\_strategy(characteristics); // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿┘ç█î┘å┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪¬┘ê╪╡█î┘ç┘ç╪º█î AI switch (strategy.approach) { case serialization\_approach::binary\_optimized: generate\_binary\_optimized\_serialization(target); break; case serialization\_approach::json\_pretty: generate\_human\_readable\_json(target); break; case serialization\_approach::compressed: generate\_compressed\_serialization(target); break; } // ╪º╪╣┘à╪º┘ä ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ┘╛█î╪┤┘å┘ç╪º╪»█î AI apply\_ai\_optimizations(target, strategy.optimizations); } // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿╪º █î╪º╪»┌»█î╪▒█î ╪¬┘é┘ê█î╪¬█î class rl\_code\_generator { // ╪╣╪º┘à┘ä █î╪º╪»┌»█î╪▒█î ╪¬┘é┘ê█î╪¬█î ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» rl\_agent code\_generation\_agent\_; public: // █î╪º╪»┌»█î╪▒█î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ╪¿┘ç█î┘å┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪º╪▓ ╪¿╪º╪▓╪«┘ê╪▒╪» void train\_on\_codebase(const codebase& training\_data) { for (const auto& example : training\_data.get\_examples()) { // ╪º╪│╪¬╪«╪▒╪º╪¼ ┘ê╪╢╪╣█î╪¬ (┘à╪┤╪«╪╡╪º╪¬ ┘å┘ê╪╣) auto state = extract\_generation\_state(example.type); // ╪º┘å╪¬╪«╪º╪¿ ╪╣┘à┘ä ╪¬┘ê┘ä█î╪» ╪¬┘ê╪│╪╖ ╪╣╪º┘à┘ä auto action = code\_generation\_agent\_.select\_action(state); // ╪º╪╣┘à╪º┘ä ╪╣┘à┘ä ┘ê ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ┘╛╪º╪»╪º╪┤ (┌⌐╪º╪▒╪º█î█î╪î ╪«┘ê╪º┘å╪º█î█î╪î ┘ê ╪║█î╪▒┘ç) auto generated\_code = apply\_generation\_action(action, example.type); auto reward = evaluate\_generated\_code(generated\_code, example.expected\_behavior); // ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î ╪╣╪º┘à┘ä ╪¿╪▒ ╪º╪│╪º╪│ ┘╛╪º╪»╪º╪┤ code\_generation\_agent\_.update(state, action, reward); } } // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿┘ç█î┘å┘ç ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î █î╪º╪»┌»╪▒┘ü╪¬┘ç generated\_code generate\_optimal\_code(std::meta::info target) { auto state = extract\_generation\_state(target); auto optimal\_action = code\_generation\_agent\_.get\_optimal\_action(state); return apply\_generation\_action(optimal\_action, target); } }; } ### 9.2.2 WebAssembly ┘ê ╪º┘ç╪»╪º┘ü ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ╪¬┘ê╪│╪╣┘ç┘ç╪º█î ╪ó█î┘å╪»┘ç ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ╪¿╪▒ ╪¬┘ê┘ä█î╪» ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à ┘ê ╪│╪º╪▓┌»╪º╪▒█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘à╪▒┌⌐╪▓ ╪«┘ê╪º┘ç┘å╪» ┌⌐╪▒╪» [143]: cpp // ╪¬┘ê┘ä█î╪» ┌⌐╪» ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ namespace cross\_platform { // ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à enum class target\_platform { native\_x86\_64, native\_arm64, webassembly, gpu\_cuda, gpu\_opencl }; // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪ó┌»╪º┘ç ╪º╪▓ ┘╛┘ä╪¬┘ü╪▒┘à template<target\_platform Platform> constexpr void platform\_optimized(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à if constexpr (Platform == target\_platform::webassembly) { generate\_wasm\_optimized\_code(target); } else if constexpr (Platform == target\_platform::gpu\_cuda) { generate\_cuda\_kernels(target); } else if constexpr (Platform == target\_platform::native\_x86\_64) { generate\_simd\_optimized\_code(target); } // ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪┤╪¬╪▒┌⌐ ╪»╪▒ ┘╛┘ä╪¬┘ü╪▒┘à┘ç╪º generate\_cross\_platform\_interface(target); } // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪«╪º╪╡ WebAssembly constexpr void wasm\_optimized(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪│╪º╪▓┌»╪º╪▒ ╪¿╪º WASM generate\_wasm\_binary\_serialization(target); // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪¿╪▒╪º█î ┘à╪»┘ä ╪¡╪º┘ü╪╕┘ç WASM auto members = std::meta::data\_members\_of(target); std::meta::template\_for<members>([](auto member) { apply\_wasm\_memory\_layout\_optimization(member); }); // ╪¬┘ê┘ä█î╪» ╪▒╪º╪¿╪╖ interop WASM-JavaScript generate\_js\_binding\_interface(target); // ┌⌐┘à█î┘å┘ç╪│╪º╪▓█î ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î WASM apply\_size\_optimizations(target); } // ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘à╪¡╪º╪│╪¿╪º╪¬ GPU constexpr void gpu\_accelerated(std::meta::info target) { // ╪¬┘ê┘ä█î╪» kernel ┘ç╪º█î CUDA/OpenCL ╪¿╪▒╪º█î ╪╣┘à┘ä█î╪º╪¬ ┘à┘ê╪º╪▓█î auto methods = std::meta::member\_functions\_of(target); std::meta::template\_for<methods>([](auto method) { if (is\_parallelizable(method)) { generate\_gpu\_kernel(method); generate\_cpu\_gpu\_bridge(method); } }); // ┘à╪»█î╪▒█î╪¬ ╪¡╪º┘ü╪╕┘ç ╪¿╪▒╪º█î GPU generate\_gpu\_memory\_management(target); // ┘ç┘à┌»╪º┘à╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ CPU-GPU generate\_synchronization\_code(target); } } // ┘à╪½╪º┘ä ╪º╪│╪¬┘é╪▒╪º╪▒ ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç class $platform\_optimized<cross\_platform::target\_platform::webassembly> $gpu\_accelerated DataProcessor { std::vector<float> data; // ╪«┘ê╪»┌⌐╪º╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à ╪¬┘ê┘ä█î╪» ┘à█î┌⌐┘å╪» void process\_data() { // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î CPU ╪¿╪▒╪º█î ┘à╪¼┘à┘ê╪╣┘ç ╪»╪º╪»┘ç┘ç╪º█î ┌⌐┘ê┌å┌⌐ // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î GPU ╪¿╪▒╪º█î ┘à╪¼┘à┘ê╪╣┘ç ╪»╪º╪»┘ç┘ç╪º█î ╪¿╪▓╪▒┌» // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪¿┘ç█î┘å┘ç ╪┤╪»┘ç WASM ╪¿╪▒╪º█î ╪º╪│╪¬┘é╪▒╪º╪▒ ┘ê╪¿ } // ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç auto serialize() const { // ┘ü╪▒┘à╪¬ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à┘å╪º╪│╪¿ ┘╛┘ä╪¬┘ü╪▒┘à } }; ### 9.2.3 ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌» ┘ê ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒╪º█î ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿┘ç╪¬╪▒ ╪º╪▓ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌» ┘ê ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç ╪¬┌⌐╪º┘à┘ä ╪«┘ê╪º┘ç┘å╪» █î╪º┘ü╪¬ [144]: cpp // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌» ┘ê ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç namespace realtime { // ┘à╪┤╪«╪╡╪º╪¬ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌» struct rt\_constraints { std::chrono::nanoseconds max\_execution\_time; size\_t max\_memory\_usage; bool deterministic\_timing\_required; priority\_level task\_priority; }; // ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪ó┌»╪º┘ç ╪º╪▓ ╪¿┘ä╪º╪»╪▒┘å┌» template<rt\_constraints Constraints> constexpr void realtime\_entity(std::meta::info target) { // ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘à┘å╪º╪│╪¿ ╪¿┘ê╪»┘å ╪¿┘ä╪º╪»╪▒┘å┌» static\_assert(validate\_rt\_suitability<Constraints>(target), "Type not suitable for real-time constraints"); // ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘é╪╖╪╣█î if constexpr (Constraints.deterministic\_timing\_required) { generate\_deterministic\_implementations(target); eliminate\_dynamic\_memory\_allocation(target); } // ╪¬╪«╪╡█î╪╡ ╪º╪│╪¬╪«╪▒ ╪¡╪º┘ü╪╕┘ç generate\_memory\_pool\_allocators(target, Constraints.max\_memory\_usage); // ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪¿╪»┘ê┘å ┘é┘ü┘ä ╪»╪▒ ╪╡┘ê╪▒╪¬ ╪º┘à┌⌐╪º┘å generate\_lockfree\_data\_structures(target); // ┘å╪╕╪º╪▒╪¬ ╪¿┘ä╪º╪»╪▒┘å┌» if constexpr (debug\_mode) { generate\_timing\_assertions(target, Constraints.max\_execution\_time); } } // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪│█î╪│╪¬┘à ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç constexpr void embedded\_optimized(std::meta::info target) { // ┌⌐┘à█î┘å┘ç╪│╪º╪▓█î ╪▒╪» ┘╛╪º█î ╪¡╪º┘ü╪╕┘ç apply\_memory\_optimizations(target); // ╪¡╪░┘ü overhead ╪¬╪º╪¿╪╣ ┘à╪¼╪º╪▓█î ╪»╪▒ ╪╡┘ê╪▒╪¬ ╪º┘à┌⌐╪º┘å devirtualize\_methods(target); // ╪¬┘ê┘ä█î╪» ╪¼╪»╪º┘ê┘ä ╪¼╪│╪¬╪¼┘ê█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä generate\_constexpr\_lookup\_tables(target); // ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ┘ü┘ä╪┤ apply\_flash\_optimizations(target); } // ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪│█î╪│╪¬┘à ╪¿╪¡╪▒╪º┘å█î ╪º█î┘à┘å█î constexpr void safety\_critical(std::meta::info target) { // ╪¬┘ê┘ä█î╪» ╪¿╪▒╪▒╪│█î┘ç╪º█î ╪º█î┘à┘å█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º generate\_bounds\_checking(target); generate\_null\_pointer\_checks(target); generate\_overflow\_checking(target); // ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¬╪ú█î█î╪» ╪▒╪│┘à█î generate\_verification\_annotations(target); // ╪º┘ü╪▓┘ê┘å┌»█î ╪¿╪▒╪º█î ╪¬╪¡┘à┘ä ╪«╪╖╪º generate\_redundant\_computations(target); // ╪º┘å╪╖╪¿╪º┘é ┌»┘ê╪º┘ç█î┘å╪º┘à┘ç ensure\_certification\_compliance(target); } } // ┘à╪½╪º┘ä ╪│█î╪│╪¬┘à ╪¿┘ä╪º╪»╪▒┘å┌» constexpr rt\_constraints sensor\_constraints{ .max\_execution\_time = std::chrono::microseconds(100), .max\_memory\_usage = 1024, // ╪¿╪º█î╪¬ .deterministic\_timing\_required = true, .task\_priority = priority\_level::high }; class $realtime\_entity<sensor\_constraints> $embedded\_optimized $safety\_critical SensorData { float temperature; float pressure; std::chrono::steady\_clock::time\_point timestamp; // ┘ç┘à┘ç ┘à╪¬╪»┘ç╪º ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌» ╪¿┘ç█î┘å┘ç ╪┤╪»┘å╪» void update\_readings(float temp, float press) { // ╪▓┘à╪º┘å ╪º╪¼╪▒╪º█î ┘é╪╖╪╣█î ┘ê ┘à╪¡╪»┘ê╪» // ╪¿╪»┘ê┘å ╪¬╪«╪╡█î╪╡ ╪¡╪º┘ü╪╕┘ç ┘╛┘ê█î╪º // ╪¿╪▒╪▒╪│█î┘ç╪º█î ╪º█î┘à┘å█î ╪┤╪º┘à┘ä ╪┤╪»┘ç } }; ## 9.3 ╪¬┌⌐╪º┘à┘ä ╪º╪¿╪▓╪º╪▒┘ç╪º ┘ê IDE ### 9.3.1 ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪»█î╪¿╪º┌» ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» ╪ó█î┘å╪»┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘╛█î┌å█î╪»┘ç╪º█î ╪º╪▓ ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪▒╪º╪ª┘ç ╪«┘ê╪º┘ç┘å╪» ╪»╪º╪» [145]: cpp // ╪▓█î╪▒╪│╪º╪«╪¬ ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪»█î╪¿╪º┌» ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ namespace debug\_support { // █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç debugger class reflection\_debugger { public: // ╪¿╪º╪▓╪▒╪│█î ╪¬╪╡┘ê█î╪▒█î meta-object debug\_visualization visualize\_meta\_object(std::meta::info meta\_obj) { debug\_visualization viz; // ╪º█î╪¼╪º╪» ┘å┘à╪º█î ╪»╪▒╪«╪¬█î ╪¬╪╣╪º┘à┘ä█î ╪│┘ä╪│┘ä┘ç ┘à╪▒╪º╪¬╪¿ meta-object viz.root = create\_meta\_object\_tree\_node(meta\_obj); // ╪º╪╢╪º┘ü┘ç ┌⌐╪▒╪»┘å ┘╛╪º┘å┘ä┘ç╪º█î ╪«╪º╪╡█î╪¬ meta-object viz.properties = extract\_meta\_object\_properties(meta\_obj); // ┘å╪┤╪º┘å ╪»╪º╪»┘å ╪▒┘ê╪º╪¿╪╖ ╪¿╪º ╪│╪º█î╪▒ meta-object ┘ç╪º viz.relationships = find\_meta\_object\_relationships(meta\_obj); return viz; } // ╪»█î╪¿╪º┌» ┘é╪»┘à ╪¿┘ç ┘é╪»┘à ┌⌐╪º╪▒╪¿╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ debug\_session debug\_metaclass\_application( const metaclass\_application& application) { debug\_session session; // ╪¬┘å╪╕█î┘à ┘å┘é╪º╪╖ ╪┤┌⌐╪│╪¬ ╪»╪▒ ┘ç╪▒ ┘à╪▒╪¡┘ä┘ç ╪¬┘ê┘ä█î╪» session.breakpoints = create\_generation\_breakpoints(application); // ╪▒╪»█î╪º╪¿█î ╪¬╪║█î█î╪▒╪º╪¬ ┘ê╪╢╪╣█î╪¬ meta-object session.state\_tracker = create\_meta\_object\_state\_tracker(application); // ╪¬╪¼╪│┘à ┘ü╪▒╪ó█î┘å╪» ╪¬┘ê┘ä█î╪» ┌⌐╪» session.generation\_visualizer = create\_generation\_visualizer(application); return session; } // ╪»█î╪¿╪º┌» ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º void debug\_runtime\_reflection(const runtime\_reflection\_context& context) { // ┘å╪┤╪º┘å ╪»╪º╪»┘å ╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê╪╣ ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘à┘ê╪¼┘ê╪» display\_runtime\_types(context); // ╪¿╪º╪▓╪▒╪│█î ╪¬╪╣╪º┘à┘ä█î ╪╣╪╢┘ê enable\_interactive\_member\_inspection(context); // ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪¬╪» ┘╛┘ê█î╪º ╪º╪▓ debugger enable\_debugger\_method\_invocation(context); } }; // ╪▒╪»█î╪º╪¿█î ╪¬┘ê┘ä█î╪» ┌⌐╪» class generation\_tracer { std::vector<generation\_step> trace\_; public: void record\_generation\_step(const generation\_step& step) { trace\_.push\_back(step); // ╪¬╪¼╪│┘à ╪▒╪»█î╪º╪¿█î ╪¿┘ä╪º╪»╪▒┘å┌» if (debugger\_attached()) { send\_trace\_update\_to\_debugger(step); } } // ╪¿╪º╪▓┘╛╪«╪┤ ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿╪▒╪º█î ╪»█î╪¿╪º┌» void replay\_generation(const replay\_options& options) { for (const auto& step : trace\_) { if (options.should\_replay\_step(step)) { replay\_generation\_step(step); if (options.interactive\_mode) { wait\_for\_debugger\_continuation(); } } } } }; } ### 9.3.2 ╪¬┘é┘ê█î╪¬┘ç╪º█î IDE ┘à╪¡█î╪╖┘ç╪º█î ╪¬┘ê╪│╪╣┘ç █î┌⌐┘╛╪º╪▒┌å┘ç ╪¿╪▒╪º█î ╪º╪▒╪º╪ª┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¼╪º┘à╪╣ ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┌⌐╪º┘à┘ä ╪«┘ê╪º┘ç┘å╪» █î╪º┘ü╪¬ [146]: cpp // ┘à╪┤╪«╪╡╪º╪¬ ╪¬┘é┘ê█î╪¬ IDE namespace ide\_enhancements { // ╪¬┌⌐┘à█î┘ä ┌⌐╪» ┘ç┘ê╪┤┘à┘å╪» ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ class reflection\_intellisense { public: // ╪¬┌⌐┘à█î┘ä┘ç╪º█î meta-object ╪ó┌»╪º┘ç ╪º╪▓ ╪▓┘à█î┘å┘ç completion\_list get\_meta\_object\_completions( const code\_context& context, const std::meta::info& meta\_obj) { completion\_list completions; // ╪╣┘à┘ä█î╪º╪¬ ┘à┘ê╪¼┘ê╪» ╪¿╪▒ ╪º╪│╪º╪│ ┘å┘ê╪╣ meta-object auto operations = get\_available\_operations(meta\_obj); for (const auto& op : operations) { completions.add\_operation\_completion(op); } // ╪¬┌⌐┘à█î┘ä┘ç╪º█î ╪»╪│╪¬╪▒╪│█î ╪╣╪╢┘ê if (is\_type\_meta\_object(meta\_obj)) { auto members = get\_type\_members(meta\_obj); for (const auto& member : members) { completions.add\_member\_completion(member); } } return completions; } // ╪¬┌⌐┘à█î┘ä┘ç╪º█î ┘é╪º┘ä╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ completion\_list get\_metaclass\_completions(const type\_context& context) { completion\_list completions; // ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘à┘ê╪¼┘ê╪» ╪¿╪▒ ╪º╪│╪º╪│ ┘à╪┤╪«╪╡╪º╪¬ ┘å┘ê╪╣ auto suitable\_metaclasses = find\_suitable\_metaclasses(context); for (const auto& metaclass : suitable\_metaclasses) { completions.add\_metaclass\_completion(metaclass); } return completions; } }; // ┘╛█î╪┤┘å┘à╪º█î╪┤ ╪▓┘å╪»┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» class live\_generation\_preview { public: // ┘å┘à╪º█î╪┤ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿┘ä╪º╪»╪▒┘å┌» ┘ç┘à╪▒╪º┘ç ╪¿╪º ╪¬╪º█î┘╛ ┌⌐╪º╪▒╪¿╪▒ generated\_code\_preview get\_live\_preview( const partial\_metaclass\_application& partial\_app) { // ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿╪▒ ╪º╪│╪º╪│ ┘ê╪╢╪╣█î╪¬ ┘ü╪╣┘ä█î auto generated = simulate\_code\_generation(partial\_app); // ╪¿╪▒╪¼╪│╪¬┘ç╪│╪º╪▓█î ╪¬┘ü╪º┘ê╪¬┘ç╪º ╪º╪▓ ┘╛█î╪┤┘å┘à╪º█î╪┤ ┘é╪¿┘ä█î auto differences = compute\_generation\_differences( generated, previous\_preview\_); previous\_preview\_ = generated; return generated\_code\_preview{ .generated\_code = generated, .differences = differences, .compilation\_status = check\_compilation\_status(generated) }; } // ╪¬┘å╪╕█î┘à ╪¬╪╣╪º┘à┘ä█î ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ void adjust\_metaclass\_parameters( const parameter\_adjustment& adjustment) { // ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î ┌⌐╪º╪▒╪¿╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î ╪¼╪»█î╪» update\_metaclass\_application(adjustment); // ╪¬┘ê┘ä█î╪» ┘à╪¼╪»╪» ┘╛█î╪┤┘å┘à╪º█î╪┤ auto new\_preview = get\_live\_preview(current\_application\_); // ╪¿┘ç╪▒┘ê╪▓╪▒╪│╪º┘å█î ┘å┘à╪º█î╪┤ IDE update\_preview\_display(new\_preview); } private: generated\_code previous\_preview\_; partial\_metaclass\_application current\_application\_; }; // ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪│╪º╪«╪¬╪º╪▒█î ╪¿╪▒╪º█î ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿ class reflection\_refactoring { public: // ╪¬╪║█î█î╪▒ ┘å╪º┘à ╪º█î┘à┘å ╪º╪╣╪╢╪º█î ╪¿╪º╪▓╪¬╪º╪¿█î refactoring\_plan plan\_member\_rename( const member\_reference& member, const std::string& new\_name) { refactoring\_plan plan; // ┘╛█î╪»╪º ┌⌐╪▒╪»┘å ┘ç┘à┘ç ┘à╪▒╪º╪¼╪╣ ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ auto reflection\_refs = find\_reflection\_references(member); for (const auto& ref : reflection\_refs) { plan.add\_change(create\_reflection\_reference\_update(ref, new\_name)); } // ┘╛█î╪»╪º ┌⌐╪▒╪»┘å ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ┌⌐┘ç ╪º╪▓ ╪╣╪╢┘ê ╪º╪│╪¬┘ü╪º╪»┘ç ┘à█î┌⌐┘å╪» auto generated\_refs = find\_generated\_code\_references(member); for (const auto& ref : generated\_refs) { plan.add\_regeneration\_request(ref.containing\_type); } return plan; } // ╪º╪│╪¬╪«╪▒╪º╪¼ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪▓ ╪º┘ä┌»┘ê┘ç╪º█î ╪¬┌⌐╪▒╪º╪▒█î refactoring\_plan extract\_metaclass( const std::vector<type\_reference>& similar\_types) { // ╪¬╪¡┘ä█î┘ä ╪º┘ä┌»┘ê┘ç╪º█î ┘à╪┤╪¬╪▒┌⌐ auto common\_patterns = analyze\_common\_patterns(similar\_types); // ╪¬┘ê┘ä█î╪» ┘é╪º┘ä╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ auto metaclass\_template = generate\_metaclass\_template(common\_patterns); // ╪¿╪▒┘å╪º┘à┘ç╪▒█î╪▓█î ┌⌐╪º╪▒╪¿╪▒╪» ╪¿╪▒╪º█î ╪º┘å┘ê╪º╪╣ ┘à┘ê╪¼┘ê╪» refactoring\_plan plan; plan.add\_metaclass\_creation(metaclass\_template); for (const auto& type : similar\_types) { plan.add\_metaclass\_application(type, metaclass\_template); } return plan; } }; } ### 9.3.3 ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪«╪╡╪╡█î ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪╢╪▒┘ê╪▒█î ╪«┘ê╪º┘ç┘å╪» ╪¿┘ê╪» [147]: cpp // ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ namespace performance\_tools { // ╪¬╪¡┘ä█î┘ä┌»╪▒ ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä class compilation\_analyzer { std::vector<compilation\_metric> metrics\_; public: // ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪¬╪ú╪½█î╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪▒┘ê█î ┌⌐╪º┘à┘╛╪º█î┘ä compilation\_impact measure\_metaclass\_impact( const metaclass\_application& application) { compilation\_impact impact; // ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º ┘ê ╪¿╪»┘ê┘å ┘à╪¬╪º┌⌐┘ä╪º╪│ auto baseline\_time = measure\_baseline\_compilation(application.target\_type); auto metaclass\_time = measure\_metaclass\_compilation(application); impact.time\_overhead = metaclass\_time - baseline\_time; impact.memory\_overhead = measure\_memory\_overhead(application); impact.binary\_size\_impact = measure\_binary\_size\_impact(application); return impact; } // ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ reflection\_performance\_profile profile\_reflection\_queries( const std::vector<reflection\_query>& queries) { reflection\_performance\_profile profile; for (const auto& query : queries) { auto query\_metrics = measure\_query\_performance(query); profile.add\_query\_metrics(query, query\_metrics); } // ╪┤┘å╪º╪│╪º█î█î ┌»┘ä┘ê┌»╪º┘ç┘ç╪º█î ┌⌐╪º╪▒╪º█î█î profile.bottlenecks = identify\_performance\_bottlenecks(profile); // ┘╛█î╪┤┘å┘ç╪º╪» ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º profile.optimization\_suggestions = suggest\_optimizations(profile); return profile; } }; // ┘╛╪▒┘ê┘ü╪º█î┘ä╪▒ ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º class runtime\_profiler { public: // ┘╛╪▒┘ê┘ü╪º█î┘ä ┌⌐╪º╪▒╪º█î█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç runtime\_profile profile\_generated\_code( const generated\_code\_execution& execution) { runtime\_profile profile; // ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º█î ┘à╪¬╪»┘ç╪º█î ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç profile.method\_timings = measure\_method\_timings(execution); // ╪º┘ä┌»┘ê┘ç╪º█î ╪¬╪«╪╡█î╪╡ ╪¡╪º┘ü╪╕┘ç profile.allocation\_patterns = analyze\_allocation\_patterns(execution); // ┌⌐╪º╪▒╪º█î█î ┌⌐╪┤ profile.cache\_metrics = measure\_cache\_performance(execution); return profile; } // ┘à┘é╪º█î╪│┘ç ┌⌐╪º╪▒╪º█î█î ╪¿╪º ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪»╪│╪¬█î performance\_comparison compare\_with\_manual( const generated\_implementation& generated, const manual\_implementation& manual) { performance\_comparison comparison; // ┘à┘é╪º█î╪│┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º comparison.execution\_time\_ratio = measure\_execution\_time(generated) / measure\_execution\_time(manual); // ┘à┘é╪º█î╪│┘ç ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç comparison.memory\_usage\_ratio = measure\_memory\_usage(generated) / measure\_memory\_usage(manual); // ┘à┘é╪º█î╪│┘ç ╪º┘å╪»╪º╪▓┘ç ┌⌐╪» comparison.code\_size\_ratio = measure\_code\_size(generated) / measure\_code\_size(manual); return comparison; } }; // ┘à┘ê╪¬┘ê╪▒ ╪¬┘ê╪╡█î┘ç ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î class optimization\_engine { public: // ╪¬╪¡┘ä█î┘ä ╪»╪º╪»┘ç┘ç╪º█î ┌⌐╪º╪▒╪º█î█î ┘ê ┘╛█î╪┤┘å┘ç╪º╪» ╪¿┘ç╪¿┘ê╪»┘ç╪º optimization\_recommendations analyze\_performance( const performance\_data& data) { optimization\_recommendations recommendations; // ╪┤┘å╪º╪│╪º█î█î ┘à╪│█î╪▒┘ç╪º█î ╪»╪º╪║ ╪»╪▒ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç auto hot\_paths = identify\_hot\_paths(data); for (const auto& path : hot\_paths) { recommendations.add\_hot\_path\_optimization(path); } // ┘╛█î╪┤┘å┘ç╪º╪» ╪¬┘å╪╕█î┘à╪º╪¬ ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ auto parameter\_suggestions = suggest\_parameter\_adjustments(data); recommendations.add\_parameter\_suggestions(parameter\_suggestions); // ╪¬┘ê╪╡█î┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪¼╪º█î┌»╪▓█î┘å auto alternative\_metaclasses = suggest\_alternative\_metaclasses(data); recommendations.add\_alternative\_suggestions(alternative\_metaclasses); return recommendations; } }; } ╪º█î┘å ╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç ┘╛╪¬╪º┘å╪│█î┘ä ╪╣╪╕█î┘à ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒╪º█î ╪¬╪¡┘ê┘ä ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪». ╪¬╪▒┌⌐█î╪¿ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪▓╪¿╪º┘å█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç╪î ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¿┘ç╪¿┘ê╪» █î╪º┘ü╪¬┘ç╪î ┘ê █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º█î ┘å┘ê╪╕┘ç┘ê╪▒ ┘ê╪╣╪»┘ç ╪º█î┘å ╪▒╪º ┘à█î╪»┘ç╪» ┌⌐┘ç C++ ╪▒╪º ╪¡╪¬█î ┘é╪»╪▒╪¬┘à┘å╪»╪¬╪▒ ┘ê ╪¿█î╪º┘å┌»╪▒╪¬╪▒ ┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î ╪ó┘å ╪▒╪º ╪¡┘ü╪╕ ┌⌐┘å╪». ╪¿╪º ╪¬┌⌐╪º┘à┘ä ╪º┌⌐┘ê╪│█î╪│╪¬┘à╪î ┘à█î╪¬┘ê╪º┘å█î┘à ╪º┘å╪¬╪╕╪º╪▒ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘ü╪▓╪º█î┘å╪»┘ç ┘╛█î┌å█î╪»┘ç ╪º█î┘å ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º ╪▒╪º ╪»╪▒ ┘ç┘à┘ç ╪¡┘ê╪▓┘ç┘ç╪º█î ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤█î┘à. ΓÇö *[┘à╪▒╪º╪¼╪╣ 139-147 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º╪î ╪¬┘ê╪│╪╣┘ç ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º ML╪î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç╪î ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿┘ä╪º╪»╪▒┘å┌»╪î ╪▓█î╪▒╪│╪º╪«╪¬ ╪»█î╪¿╪º┌»╪î ╪¬┘é┘ê█î╪¬┘ç╪º█î IDE ┘ê ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # 10. ┘å╪¬█î╪¼┘ç┌»█î╪▒█î ## 10.1 ╪«┘ä╪º╪╡┘ç █î╪º┘ü╪¬┘ç┘ç╪º█î ┌⌐┘ä█î╪»█î ╪º█î┘å ┘à╪╖╪º┘ä╪╣┘ç ╪¼╪º┘à╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ┘╛╪¬╪º┘å╪│█î┘ä ╪¬╪¡┘ê┘ä╪ó┘ü╪▒█î┘å ╪ó┘å┘ç╪º ╪¿╪▒╪º█î ╪ó█î┘å╪»┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪»╪▒ C++ ╪▒╪º ┘å╪┤╪º┘å ╪»╪º╪»┘ç ╪º╪│╪¬. ╪º╪▓ ╪╖╪▒█î┘é ╪¬╪¡┘ä█î┘ä ╪»┘é█î┘é ┌å╪º╪▒┌å┘ê╪¿ ┘ü┘å█î╪î ┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î╪î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪╣┘à┘ä█î ┘ê ╪º┘à┌⌐╪º┘å╪º╪¬ █î┌⌐┘╛╪º╪▒┌å┌»█î╪î ┌å┘å╪»█î┘å █î╪º┘ü╪¬┘ç ┌⌐┘ä█î╪»█î ╪╕┘ç┘ê╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘à╪¼┘à┘ê╪╣╪º┘ï ╪¬╪╡┘ê█î╪▒█î ╪º╪▓ ╪¬╪║█î█î╪▒ ┘╛╪º╪▒╪º╪»╪º█î┘à ╪»╪▒ ┘å╪¡┘ê┘ç ╪▒┘ê█î┌⌐╪▒╪» ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å C++ ╪¿┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¬╪▒╪│█î┘à ┘à█î┌⌐┘å┘å╪». ### 10.1.1 ╪»╪│╪¬╪º┘ê╪▒╪»┘ç╪º█î ┘ü┘å█î **╪¿┘ä┘ê╪║ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐:** API ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¬┌⌐╪º┘à┘ä ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º╪▓ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪». ╪¬╪¡┘ä█î┘ä ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪º┘å╪¬╪▓╪º╪╣ std::meta::info ┘╛╪º█î┘ç ┘à╪¡┌⌐┘à█î ╪¿╪▒╪º█î ╪»╪▒┘ê┘å┘å┌»╪▒█î ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪º╪╡┘ä ╪╡┘ü╪▒-overhead C++ ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å╪» [148]. ╪╣┘à┘ä┌»╪▒ reflexpr ┘ê ╪¬┘ê╪º╪¿╪╣ ┌⌐┘ê╪ª╪▒█î ┘à╪▒╪¬╪¿╪╖ ╪»╪│╪¬╪▒╪│█î ╪¿█î╪│╪º╪¿┘é┘ç╪º█î ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ╪¿╪»┘ê┘å ╪¼╪▒█î┘à┘ç ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç┘å╪». **┘é╪»╪▒╪¬ ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¬╪º┌⌐┘ä╪º╪│:** ╪¬╪│┘ç█î┘ä╪º╪¬ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘╛█î┌å█î╪»┘ç╪º█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘é╪¿┘ä╪º┘ï ╪║█î╪▒┘à┘à┌⌐┘å █î╪º ╪¿╪│█î╪º╪▒ ┘╛█î┌å█î╪»┘ç ╪¿╪▒╪º█î ╪»╪│╪¬█î╪º╪¿█î ╪¿┘ê╪». ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¬╪º 80% ┌⌐╪» boilerplate ╪▒╪º ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪▒╪º█î╪¼ ┘à╪º┘å┘å╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î╪î ┘å┌»╪º╪┤╪¬ ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ┘ê ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪º┘ä┌»┘ê█î observer ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐┘å┘å╪» [149]. ╪º█î┘å ╪«┘ê╪»┌⌐╪º╪▒╪│╪º╪▓█î ┘å┘ç ╪¬┘å┘ç╪º ╪▓┘à╪º┘å ╪¬┘ê╪│╪╣┘ç ╪▒╪º ┌⌐╪º┘ç╪┤ ┘à█î╪»┘ç╪» ╪¿┘ä┌⌐┘ç ╪½╪¿╪º╪¬ ┘ê ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ┌⌐╪» ╪▒╪º ┘å█î╪▓ ╪¿┘ç ╪╖┘ê╪▒ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿┘ç╪¿┘ê╪» ┘à█î╪¿╪«╪┤╪». **┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î:** ┘à╪╣█î╪º╪▒╪│┘å╪¼█î ╪¼╪º┘à╪╣ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪»╪▒ ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä (40-50% ┌⌐╪º┘ç╪┤ ╪»╪▒ ┌⌐╪»╪¿█î╪│┘ç╪º█î ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿) ╪»╪│╪¬█î╪º╪¿█î ┘à█î┌⌐┘å┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º█î █î┌⌐╪│╪º┘å ╪¿╪º ┌⌐╪» ╪»╪│╪¬┘å┘ê█î╪│ ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å┘å╪» [150]. ┘à╪»┘ä ╪º╪▒╪▓█î╪º╪¿█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¬╪╢┘à█î┘å ┘à█î┌⌐┘å╪» ┌⌐┘ç ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┘ç█î┌å overhead ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¬╪¡┘à█î┘ä ┘å┘à█î┌⌐┘å┘å╪» ┘ê ┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î C++ ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å┘å╪». ### 10.1.2 ╪º╪▒╪▓█î╪º╪¿█î ╪¬╪ú╪½█î╪▒ ╪╣┘à┘ä█î **╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç:** ╪│╪º╪▓┘à╪º┘å┘ç╪º█î█î ┌⌐┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪º┘ê┘ä█î┘ç ╪▒╪º ┘╛╪░█î╪▒┘ü╪¬┘ç╪º┘å╪» ┌»╪▓╪º╪▒╪┤ ┘╛█î╪┤╪▒┘ü╪¬┘ç╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ┘à█î╪»┘ç┘å╪». ╪¬┘ê╪º┘å╪º█î█î ╪¡╪░┘ü ╪º┘ä┌»┘ê┘ç╪º█î ┌⌐╪»┘å┘ê█î╪│█î ╪¬┌⌐╪▒╪º╪▒█î ╪º╪▓ ╪╖╪▒█î┘é ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪º╪¼╪º╪▓┘ç ┘à█î╪»┘ç╪» ╪¿╪▒ ┘à┘å╪╖┘é ╪«╪º╪╡ ╪¡┘ê╪▓┘ç ╪¬┘à╪▒┌⌐╪▓ ┌⌐┘å┘å╪» ╪¬╪º ┘å┌»╪▒╪º┘å█î┘ç╪º█î ╪▓█î╪▒╪│╪º╪«╪¬█î [151]. ╪¬╪¡┘ä█î┘ä ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º┘å╪»╪º╪▓┘ç ┌⌐╪»╪¿█î╪│ ╪▒╪º 30-60% ╪»╪▒ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪¿╪º ╪º┘ä╪▓╪º┘à╪º╪¬ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┌⌐╪º┘ç╪┤ ╪»┘ç┘å╪». **╪¿┘ç╪¿┘ê╪» ┌⌐█î┘ü█î╪¬ ┌⌐╪»:** ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪½╪¿╪º╪¬ ╪¿╪▒╪¬╪▒█î ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪»╪│╪¬┘å┘ê█î╪│ ┘å╪┤╪º┘å ┘à█î╪»┘ç╪». ╪¡╪░┘ü ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┌⌐┘╛█î-┘╛█î╪│╪¬ ┘ê ╪º╪╣┘à╪º┘ä ╪«┘ê╪»┌⌐╪º╪▒ ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º ╪º╪▓ ╪╖╪▒█î┘é ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à┘å╪¼╪▒ ╪¿┘ç ╪¿╪º┌»┘ç╪º█î ┌⌐┘à╪¬╪▒ ┘ê ╪¿┘ç╪¿┘ê╪» ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ┘à█î╪┤┘ê╪» [152]. ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐ 70% ┘à╪│╪º╪ª┘ä ┌⌐█î┘ü█î╪¬ ┌⌐╪» ┌⌐┘à╪¬╪▒█î ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┌»╪▓╪º╪▒╪┤ ┘à█î╪»┘ç┘å╪». **┘à┘ä╪º╪¡╪╕╪º╪¬ ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î:** ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à┘ü╪º┘ç█î┘à ╪¼╪»█î╪»█î ╪▒╪º ┘à╪╣╪▒┘ü█î ┘à█î┌⌐┘å┘å╪» ┌⌐┘ç ┘å█î╪º╪▓ ╪¿┘ç ╪ó┘à┘ê╪▓╪┤ ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪»╪º╪▒┘å╪»╪î ╪¬╪¡┘ä█î┘ä ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î █î╪º╪»┌»█î╪▒█î ╪│╪▒█î╪╣╪º┘ï ╪│┘ê╪» ┘à█î╪»┘ç╪». ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å█î ┌⌐┘ç ╪¿╪º ┘à┘ü╪º┘ç█î┘à ┘à╪»╪▒┘å C++ ╪ó╪┤┘å╪º ┘ç╪│╪¬┘å╪» ┘à╪╣┘à┘ê┘ä╪º┘ï ╪╕╪▒┘ü 2-3 ┘ç┘ü╪¬┘ç █î╪º╪»┌»█î╪▒█î ┘à╪¬┘à╪▒┌⌐╪▓ ╪¿┘ç ┘à┘ç╪º╪▒╪¬ ┘à█î╪▒╪│┘å╪» [153]. ┘ê╪╢┘ê╪¡ ┘à┘ü┘ç┘ê┘à█î ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ┘à┘é╪º█î╪│┘ç ╪¿╪º ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪»╪▒ ┘ê╪º┘é╪╣ ╪¿╪º╪▒ █î╪º╪»┌»█î╪▒█î ┌⌐┘ä█î ╪¿╪▒╪º█î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪▒╪º ┌⌐╪º┘ç╪┤ ┘à█î╪»┘ç╪». ## 10.2 ┘╛█î╪º┘à╪»┘ç╪º ╪¿╪▒╪º█î ╪º┌⌐┘ê╪│█î╪│╪¬┘à C++ ### 10.2.1 ╪º┘å┘é┘ä╪º╪¿ ╪¬┘ê╪│╪╣┘ç ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç **┘à╪╣┘à╪º╪▒█î ┌å╪º╪▒┌å┘ê╪¿:** ┘à╪╣╪▒┘ü█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪º╪│╪º╪│╪º┘ï ┘å╪¡┘ê┘ç ╪╖╪▒╪º╪¡█î ┘ê ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î C++ ╪▒╪º ╪¬╪║█î█î╪▒ ┘à█î╪»┘ç╪». ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î ╪ó█î┘å╪»┘ç ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ╪▒┘ê█î┌⌐╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│-╪º┘ê┘ä ╪▒╪º ╪º╪¬╪«╪º╪░ ╪«┘ê╪º┘ç┘å╪» ┌⌐╪▒╪» ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪«╪º╪╡ ╪¡┘ê╪▓┘ç ╪▒╪º ╪¿┘ç ╪¼╪º█î ╪▒╪º╪¿╪╖┘ç╪º█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪º╪▒╪º╪ª┘ç ╪«┘ê╪º┘ç┘å╪» ╪»╪º╪» [154]. ╪º█î┘å ╪¬╪║█î█î╪▒ ┘ê╪╣╪»┘ç ╪º█î┘å ╪▒╪º ┘à█î╪»┘ç╪» ┌⌐┘ç ╪╣┘à┘ä┌⌐╪▒╪» ┘╛█î╪┤╪▒┘ü╪¬┘ç ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪▒╪º ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┌⌐╪º╪▒╪¿╪▒╪» ╪»╪▒ ╪»╪│╪¬╪▒╪│╪¬╪▒ ┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪º┘å╪╣ ╪¬╪«╪╡╪╡ ╪▒╪º ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ┘╛█î┌å█î╪»┘ç ┌⌐╪º┘ç╪┤ ╪»┘ç╪». **╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ê ┘╛╪º█î╪»╪º╪▒█î:** ╪¬╪¡┘ä█î┘ä ┘à╪º ╪º╪▓ ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º█î╪¼╪º╪» ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¼┘ç╪º┘å█î ┘ê ╪¿╪º ┌⌐╪º╪▒╪º█î█î ╪¿╪º┘ä╪º ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪». ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î█î ┘à╪º┘å┘å╪» reflection\_serializer ┘╛█î╪┤┘å┘ç╪º╪»█î ┘à╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒╪º█î ┘ç╪▒ ┘å┘ê╪╣ ╪¿╪º╪▓╪¬╪º╪¿█î ╪¿╪»┘ê┘å ┘å█î╪º╪▓ ╪¿┘ç ┘╛█î┌⌐╪▒╪¿┘å╪»█î ╪»╪│╪¬█î █î╪º ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪º╪▒╪º╪ª┘ç ╪»┘ç┘å╪» [155]. ╪º█î┘å ┘é╪º╪¿┘ä█î╪¬ ┘╛█î╪º┘à╪»┘ç╪º█î ╪╣┘à█î┘é█î ╪¿╪▒╪º█î ╪¬╪¿╪º╪»┘ä ╪»╪º╪»┘ç╪î ┘╛╪º█î╪»╪º╪▒█î ┘ê ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┘à╪¡╪º╪│╪¿╪º╪¬ ╪¬┘ê╪▓█î╪╣ ╪┤╪»┘ç ╪»╪º╪▒╪». **╪▒╪º╪¿╪╖ ┌⌐╪º╪▒╪¿╪▒█î ┘ê ╪º╪¬╪╡╪º┘ä:** ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪º╪¬╪╡╪º┘ä ╪«┘ê╪»┌⌐╪º╪▒ ╪«╪º╪╡█î╪¬ ┘å╪┤╪º┘å ╪»╪º╪»┘ç ╪┤╪»┘ç ╪»╪▒ ┘à╪╖╪º┘ä╪╣┘ç ┘à┘ê╪▒╪»█î ┌å╪º╪▒┌å┘ê╪¿ GUI ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪│█î╪º╪▒ ┘à╪¡┌⌐┘à╪¬╪▒█î ╪¿█î┘å ┘à┘å╪╖┘é ╪¬╪¼╪º╪▒█î C++ ┘ê ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º█î ╪▒╪º╪¿╪╖ ┌⌐╪º╪▒╪¿╪▒█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ╪«┘ê╪º┘ç╪» ┌⌐╪▒╪». ╪¬┘ê╪º┘å╪º█î█î ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪» ╪º╪¬╪╡╪º┘ä ┘à┘å╪¿╪╣ ╪º╪╡┘ä█î ┘╛█î┌å█î╪»┌»█î ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪»╪│┌⌐╪¬╪º┘╛ ┘ê ┘ê╪¿ ╪▒╪º ╪¡╪░┘ü ┘à█î┌⌐┘å╪» [156]. ### 10.2.2 ╪¬┌⌐╪º┘à┘ä ╪º┌⌐┘ê╪│█î╪│╪¬┘à ╪º╪¿╪▓╪º╪▒ **╪¬┘é┘ê█î╪¬ ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐:** ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿█î┘å╪┤ ╪¿█î╪│╪º╪¿┘é┘ç╪º█î ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒ ┘ê ┘é╪╡╪» ╪¿╪▒┘å╪º┘à┘ç ╪¿┘ç ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». ╪º╪¿╪▓╪º╪▒┘ç╪º ╪º┌⌐┘å┘ê┘å ┘à█î╪¬┘ê╪º┘å┘å╪» ┘å┘ç ╪¬┘å┘ç╪º ╪│╪º╪«╪¬╪º╪▒ ┘å╪¡┘ê█î ┌⌐╪» ╪¿┘ä┌⌐┘ç ╪▒┘ê╪º╪¿╪╖ ┘à╪╣┘å╪º█î█î ┌⌐╪»┌»╪░╪º╪▒█î ╪┤╪»┘ç ╪»╪▒ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘ê ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¬╪¡┘ä█î┘ä ┌⌐┘å┘å╪» [157]. ╪º█î┘å ┘é╪º╪¿┘ä█î╪¬ ╪¬╪¡┘ä█î┘ä ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¬╪┤╪«█î╪╡ ╪¿╪º┌» ┘╛█î┌å█î╪»┘ç╪¬╪▒╪î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪º╪▒╪º█î█î ┘ê ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪│╪º╪«╪¬╪º╪▒█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪». **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪│█î╪│╪¬┘à ╪│╪º╪«╪¬:** ┘à╪º┘ç█î╪¬ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪«┘ê╪¿ ╪¿╪º ┘à╪╣┘à╪º╪▒█î┘ç╪º█î ┘à╪»╪▒┘å ╪│█î╪│╪¬┘à ╪│╪º╪«╪¬ ┘ç┘à╪º┘ç┘å┌» ╪º╪│╪¬. ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪│╪º╪«╪¬ ┘à█î╪¬┘ê╪º┘å┘å╪» ╪º╪▓ ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î ┘à╪ñ╪½╪▒╪¬╪▒ ┘ê ╪º╪▒╪º╪ª┘ç ╪▒╪»█î╪º╪¿█î ┘ê╪º╪¿╪│╪¬┌»█î ╪¿┘ç╪¬╪▒ ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å┘å╪» [158]. ╪¬╪¡┘ä█î┘ä ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪│█î╪│╪¬┘à┘ç╪º█î ╪│╪º╪«╪¬ ╪ó┌»╪º┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å┘å╪» 20-30% ╪▓┘à╪º┘å ╪│╪º╪«╪¬ ╪│╪▒█î╪╣╪¬╪▒ ╪»╪▒ ┌⌐╪»╪¿█î╪│┘ç╪º█î ╪¿╪▓╪▒┌» ╪»╪│╪¬█î╪º╪¿█î ┌⌐┘å┘å╪». **╪¬┘ê┘ä█î╪» ┘à╪│╪¬┘å╪»╪º╪¬:** ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à╪│╪¬┘å╪»╪º╪¬ ╪¼╪º┘à╪╣ API ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘å┘ç ╪¬┘å┘ç╪º ╪┤╪º┘à┘ä ╪¬┘ê╪╢█î╪¡╪º╪¬ ╪▒╪º╪¿╪╖ ╪¿┘ä┌⌐┘ç ┘é╪▒╪º╪▒╪»╪º╪»┘ç╪º█î ╪▒┘ü╪¬╪º╪▒█î ┌⌐╪»┌»╪░╪º╪▒█î ╪┤╪»┘ç ╪»╪▒ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º╪┤╪». ╪º█î┘å ┘à╪│╪¬┘å╪»╪º╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ┘ç┘à█î╪┤┘ç ╪¿┘ç╪▒┘ê╪▓ ╪º╪│╪¬ ┘ê ╪¿█î┘å╪┤ ╪╣┘à█î┘é╪¬╪▒█î ╪¿┘ç ╪▒┘ü╪¬╪º╪▒ ┌⌐╪» ┘å╪│╪¿╪¬ ╪¿┘ç ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪│╪¬┘å╪»╪│╪º╪▓█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [159]. ### 10.2.3 ╪¬╪ú╪½█î╪▒ ╪ó┘à┘ê╪▓╪┤█î ┘ê ┘╛╪░█î╪▒╪┤ **╪¬╪»╪▒█î╪│ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î:** ╪¿╪º╪▓╪¬╪º╪¿ ┘å┘é╪╖┘ç ┘ê╪▒┘ê╪» ╪┤┘ç┘ê╪»█î╪¬╪▒█î ╪¿┘ç ┘à┘ü╪º┘ç█î┘à ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++ ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪». ┘à╪º┘ç█î╪¬ ╪º╪╣┘ä╪º┘å█î ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒┌⌐ ┘ê ╪º╪╣┘à╪º┘ä ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪▒╪º ╪¿╪▒╪º█î ╪»╪º┘å╪┤╪¼┘ê█î╪º┘å ┘ê ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┘à╪¿╪¬╪»█î ╪ó╪│╪º┘å╪¬╪▒ ┘à█î┌⌐┘å╪» [160]. ┘à╪ñ╪│╪│╪º╪¬ ╪ó┘à┘ê╪▓╪┤█î ┌»╪▓╪º╪▒╪┤ ┘à█î╪»┘ç┘å╪» ┌⌐┘ç ╪»╪º┘å╪┤╪¼┘ê█î╪º┘å ┘à┘ü╪º┘ç█î┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º 40% ╪│╪▒█î╪╣╪¬╪▒ ╪º╪▓ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪»╪▒┌⌐ ┘à█î┌⌐┘å┘å╪». **╪º┘ä┌»┘ê┘ç╪º█î ┘╛╪░█î╪▒╪┤ ╪╡┘å╪╣╪¬█î:** ┘╛╪░█î╪▒┘å╪»┌»╪º┘å ╪º┘ê┘ä█î┘ç ╪»╪▒ ╪╡┘å╪º█î╪╣ ╪¿╪¡╪▒╪º┘å█î ┌⌐╪º╪▒╪º█î█î (╪¿╪º╪▓█î╪î ╪«╪»┘à╪º╪¬ ┘à╪º┘ä█î╪î ╪│█î╪│╪¬┘à┘ç╪º█î ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç) ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪º ┘à┘ê┘ü┘é█î╪¬ ╪»╪▒ ┘à╪¡█î╪╖┘ç╪º█î ╪¬┘ê┘ä█î╪»█î ┘à╪│╪¬┘é╪▒ ╪┤┘ê┘å╪». ╪╢┘à╪º┘å╪¬ ╪╡┘ü╪▒-overhead ┘ê ┘à╪»┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ┘é╪╖╪╣█î ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º ╪▒╪º ╪¿╪▒╪º█î ┘à┘ê╪º╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç╪º█î ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘╛┘ê█î╪º█î ╪│┘å╪¬█î ╪║█î╪▒┘é╪º╪¿┘ä ┘é╪¿┘ê┘ä ╪«┘ê╪º┘ç╪» ╪¿┘ê╪» ┘à┘å╪º╪│╪¿ ┘à█î┌⌐┘å╪» [161]. **╪┤╪¬╪º╪¿ ┘à┘å╪º╪¿╪╣ ╪¿╪º╪▓:** ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿┘ê╪»┘å ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪¡╪º┘ä ╪¡╪º╪╢╪▒ ┘å┘ê╪ó┘ê╪▒█î ╪»╪▒ ╪¼╪º┘à╪╣┘ç ┘à┘å╪¿╪╣ ╪¿╪º╪▓ C++ ╪▒╪º ╪¬╪¡╪▒█î┌⌐ ┘à█î┌⌐┘å╪». ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º ┘ê ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪¼╪»█î╪» ╪│╪º╪«╪¬┘ç ╪┤╪»┘ç ╪¡┘ê┘ä ┘à┘ü╪º┘ç█î┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ╪¡╪º┘ä ╪╕┘ç┘ê╪▒ ┘ç╪│╪¬┘å╪» ┘ê ╪¡┘ä┘é┘ç ╪¿╪º╪▓╪«┘ê╪▒╪» ┘à╪½╪¿╪¬█î ╪º█î╪¼╪º╪» ┘à█î┌⌐┘å┘å╪» ┌⌐┘ç ┘╛╪░█î╪▒╪┤ ╪▒╪º ╪¬╪│╪▒█î╪╣ ┘à█î┌⌐┘å╪» ┘ê ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º ╪▒╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» [162]. ## 10.3 ╪¬┘ê╪╡█î┘ç┘ç╪º ╪¿╪▒╪º█î ┘à┘à╪º╪▒╪│╪º┘å ### 10.3.1 ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ┘╛╪░█î╪▒╪┤ **┘à╪╣╪▒┘ü█î ╪º┘ü╪▓╪º█î╪┤█î:** ╪│╪º╪▓┘à╪º┘å┘ç╪º ╪¿╪º█î╪» ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪¿┘ç ╪╡┘ê╪▒╪¬ ╪º┘ü╪▓╪º█î╪┤█î ┘╛╪░█î╪▒┘ü╪¬┘ç ┘ê ╪º╪▓ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┌⌐┘à ╪«╪╖╪▒ ┘ê ╪¿╪º ╪º╪▒╪▓╪┤ ╪¿╪º┘ä╪º ┘à╪º┘å┘å╪» ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘ê ╪º╪¬╪╡╪º┘ä ╪»╪º╪»┘ç ╪┤╪▒┘ê╪╣ ┌⌐┘å┘å╪». ╪¬╪¡┘ä█î┘ä ┘à╪º ┘╛█î╪┤╪▒┘ü╪¬ ┘╛╪░█î╪▒╪┤ ╪▓█î╪▒ ╪▒╪º ┘╛█î╪┤┘å┘ç╪º╪» ┘à█î┌⌐┘å╪»: 1. **┘à╪▒╪¡┘ä┘ç 1:** ╪¿╪º╪▓╪¬╪º╪¿ ┘╛╪º█î┘ç ╪¿╪▒╪º█î ╪»╪▒┘ê┘å┘å┌»╪▒█î ┘ê ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪│╪º╪»┘ç 2. **┘à╪▒╪¡┘ä┘ç 2:** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪│┘ü╪º╪▒╪┤█î ╪¿╪▒╪º█î ╪º┘ä┌»┘ê┘ç╪º█î ╪«╪º╪╡ ╪¡┘ê╪▓┘ç 3. **┘à╪▒╪¡┘ä┘ç 3:** ╪¬╪▒┌⌐█î╪¿ ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘ê ╪¬┘ê╪│╪╣┘ç ┌å╪º╪▒┌å┘ê╪¿ 4. **┘à╪▒╪¡┘ä┘ç 4:** █î┌⌐┘╛╪º╪▒┌å┌»█î ┌⌐╪º┘à┘ä ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à╪»╪▒┘å C++ ┘ê ╪º╪¿╪▓╪º╪▒┘ç╪º **╪ó┘à┘ê╪▓╪┤ ┘ê ╪¬╪╣┘ä█î┘à:** ┘╛╪░█î╪▒╪┤ ┘à┘ê┘ü┘é ┘å█î╪º╪▓ ╪¿┘ç ╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ╪»╪▒ ╪ó┘à┘ê╪▓╪┤ ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪»╪º╪▒╪». ╪│╪º╪▓┘à╪º┘å┘ç╪º ╪¿╪º█î╪» ╪¿╪▒┘å╪º┘à┘ç┘ç╪º█î ╪ó┘à┘ê╪▓╪┤█î ╪│╪º╪«╪¬╪º╪▒█î╪º┘ü╪¬┘ç ╪º╪▒╪º╪ª┘ç ╪»┘ç┘å╪» ┌⌐┘ç ╪┤╪º┘à┘ä: - ┘à┘ü╪º┘ç█î┘à ╪¿┘å█î╪º╪»█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ API - ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘ê ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º - █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ┌⌐╪»╪¿█î╪│┘ç╪º ┘ê ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ┘à┘ê╪¼┘ê╪» - ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┘ê ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪º╪▒╪º█î█î **╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ╪º╪¿╪▓╪º╪▒:** ┘à╪▓╪º█î╪º█î ┌⌐╪º┘à┘ä ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┘å┘ç╪º ╪¿╪º ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪º╪¿╪▓╪º╪▒ ┘à┘å╪º╪│╪¿ ┘à╪¡┘é┘é ┘à█î╪┤┘ê┘å╪». ╪│╪º╪▓┘à╪º┘å┘ç╪º ╪¿╪º█î╪» ╪º┘ê┘ä┘ê█î╪¬ ╪»┘ç┘å╪»: - █î┌⌐┘╛╪º╪▒┌å┌»█î IDE ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç ╪ó┌»╪º┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ - ╪¬┘é┘ê█î╪¬┘ç╪º█î ╪│█î╪│╪¬┘à ╪│╪º╪«╪¬ ╪¿╪▒╪º█î ┘╛╪▒┘ê┌ÿ┘ç┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ - ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» ┘ê ┘╛╪▒┘ê┘ü╪º█î┘ä ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç - ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐ ┌⌐┘ç ┘à╪╣┘å╪º╪┤┘å╪º╪│█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪»╪▒┌⌐ ┌⌐┘å┘å╪» ### 10.3.2 ╪▒╪º┘ç┘å┘à╪º┘ç╪º█î ╪╖╪▒╪º╪¡█î **╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│:** ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¡┘ä█î┘ä ┘à╪º ╪º╪▓ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à┘ê┘ü┘é ┘à╪¬╪º┌⌐┘ä╪º╪│╪î ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ╪▓█î╪▒ ╪▒╪º ╪¬┘ê╪╡█î┘ç ┘à█î┌⌐┘å█î┘à: **┘à╪│╪ª┘ê┘ä█î╪¬ ┘ê╪º╪¡╪»:** ┘ç╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º█î╪» ┘å┌»╪▒╪º┘å█î █î╪º ╪º┘ä┌»┘ê█î ╪«╪º╪╡█î ╪▒╪º ┘╛┘ê╪┤╪┤ ╪»┘ç╪». ╪º╪▓ ╪º█î╪¼╪º╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î █î┌⌐┘╛╪º╪▒┌å┘ç ┌⌐┘ç ╪│╪╣█î ╪»╪▒ ╪¡┘ä ┌å┘å╪»█î┘å ┘à╪│╪ª┘ä┘ç ╪║█î╪▒┘à╪▒╪¬╪¿╪╖ ╪»╪º╪▒┘å╪» ╪«┘ê╪»╪»╪º╪▒█î ┌⌐┘å█î╪» [163]. **╪¬╪▒┌⌐█î╪¿┘╛╪░█î╪▒█î:** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪╖┘ê╪▒█î ╪╖╪▒╪º╪¡█î ┌⌐┘å█î╪» ┌⌐┘ç ╪«┘ê╪¿ ╪¿╪º █î┌⌐╪»█î┌»╪▒ ┌⌐╪º╪▒ ┌⌐┘å┘å╪». ╪º╪▓ ╪▒╪º╪¿╪╖┘ç╪º█î ┘ê╪º╪╢╪¡ ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å█î╪» ┘ê ╪º╪▓ ┘ü╪▒┘ê╪╢ ╪»╪▒ ┘à┘ê╪▒╪» ╪│╪º█î╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î█î ┌⌐┘ç ┘à┘à┌⌐┘å ╪º╪│╪¬ ╪¿┘ç ┘ç┘à╪º┘å ┘å┘ê╪╣ ╪º╪╣┘à╪º┘ä ╪┤┘ê┘å╪» ╪«┘ê╪»╪»╪º╪▒█î ┌⌐┘å█î╪» [164]. **╪ó┌»╪º┘ç█î ╪º╪▓ ┌⌐╪º╪▒╪º█î█î:** ┘ç┘à█î╪┤┘ç ┘╛█î╪º┘à╪»┘ç╪º█î ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪▓┘à╪º┘å ╪º╪¼╪▒╪º█î ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪▒╪º ╪»╪▒ ┘å╪╕╪▒ ╪¿┌»█î╪▒█î╪». ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪│╪º╪»┘ç ┘ê ┘à╪│╪¬┘é█î┘à ╪▒╪º ╪¿╪▒ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪º┘ä┌»┘ê╪▒█î╪¬┘à█î ┘╛█î┌å█î╪»┘ç ╪¬╪▒╪¼█î╪¡ ╪»┘ç█î╪» [165]. **┘à╪»█î╪▒█î╪¬ ╪«╪╖╪º:** ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º█î ┘ê╪º╪╢╪¡ ┘ê ┘é╪º╪¿┘ä ╪º╪¼╪▒╪º ╪¿╪▒╪º█î ┘å┘é╪╢ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪▒╪º╪ª┘ç ╪»┘ç█î╪». ╪»╪▒ ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º█î ╪«┘ê╪¿ ╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ┌⌐┘å█î╪» ╪¬╪º ╪¬╪¼╪▒╪¿┘ç ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┘ç ╪▒╪º ╪¿┘ç╪¿┘ê╪» ╪¿╪«╪┤█î╪» [166]. **╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿:** ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┘à╪ñ╪½╪▒ ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪▒ ┌⌐╪» ┌⌐╪º╪▒╪¿╪▒╪»: **╪¬╪▒╪¼█î╪¡ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä:** ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪▒╪º ╪¿╪▒ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ┘ç╪▒ ╪¼╪º ┌⌐┘ç ╪º┘à┌⌐╪º┘å ╪»╪º╪▒╪» ╪¬╪▒╪¼█î╪¡ ╪»┘ç█î╪». ┘à╪▓╪º█î╪º█î ┌⌐╪º╪▒╪º█î█î ┘ê ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¿╪│█î╪º╪▒ ╪¿█î╪┤╪¬╪▒ ╪º╪▓ ╪▒╪º╪¡╪¬█î ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘╛┘ê█î╪º ╪»╪▒ ╪º┌⌐╪½╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º ╪º╪│╪¬ [167]. **╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ┌⌐╪┤:** ╪¿╪▒╪º█î ┘à╪¡╪º╪│╪¿╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┌»╪▒╪º┘å╪î ╪º╪▓ ┘à╪¬╪║█î╪▒┘ç╪º█î constexpr █î╪º ╪░╪«█î╪▒┘ç╪│╪º╪▓█î ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¿╪▒╪º█î ┌⌐╪┤ ┌⌐╪▒╪»┘å ┘å╪¬╪º█î╪¼ ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å█î╪». ╪º█î┘å ╪º┘ä┌»┘ê ╪«╪º╪╡┘ç ╪»╪▒ ┌⌐╪» ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ ┌⌐┘ç ┌⌐┘ê╪ª╪▒█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘à┘à┌⌐┘å ╪º╪│╪¬ ┌å┘å╪»█î┘å ╪¿╪º╪▒ ╪º╪▒╪▓█î╪º╪¿█î ╪┤┘ê┘å╪» ╪º┘ç┘à█î╪¬ ╪»╪º╪▒╪» [168]. **╪º█î┘à┘å█î ┘å┘ê╪╣:** ╪º╪▓ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪«╪╖╪º┘ç╪º█î ╪▒╪º█î╪¼ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å█î╪». ╪º╪▓ ╪¬┌⌐┘å█î┌⌐┘ç╪º█î concept ┘ê SFINAE ╪¿╪▒╪º█î ┘à╪¡╪»┘ê╪» ┌⌐╪▒╪»┘å ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪º┘å┘ê╪º╪╣ ┘à┘å╪º╪│╪¿ ╪º╪│╪¬┘ü╪º╪»┘ç ┌⌐┘å█î╪» [169]. ### 10.3.3 ╪¬╪╢┘à█î┘å ┌⌐█î┘ü█î╪¬ **╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ╪¬╪│╪¬:** ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘å█î╪º╪▓ ╪¿┘ç ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪¬╪│╪¬ ╪¬╪«╪╡╪╡█î ╪»╪º╪▒╪»: **╪¬╪│╪¬ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç:** ┘à╪¼┘à┘ê╪╣┘ç┘ç╪º█î ╪¬╪│╪¬ ╪¬┘ê╪│╪╣┘ç ╪»┘ç█î╪» ┌⌐┘ç ╪╡╪¡╪¬ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪▒╪º ╪»╪▒ ╪º┘å┘ê╪º╪╣ ┘ê╪▒┘ê╪»█î ┘ê ┘╛█î┌⌐╪▒╪¿┘å╪»█î┘ç╪º█î ┘à╪«╪¬┘ä┘ü ╪¬╪ú█î█î╪» ┌⌐┘å┘å╪». ╪¬╪│╪¬ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿┘ç ╪»┘ä█î┘ä ╪¡╪¼┘à ┌⌐╪»█î ┌⌐┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¬┘ê┘ä█î╪» ┌⌐┘å┘å╪» ╪╢╪▒┘ê╪▒█î ╪º╪│╪¬ [170]. **╪¬╪│╪¬ ╪▒┌»╪▒╪│█î┘ê┘å ┌⌐╪º╪▒╪º█î█î:** ┘å╪╕╪º╪▒╪¬ ┘à╪»╪º┘ê┘à ╪▓┘à╪º┘å┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ┌⌐╪º╪▒╪º█î█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐┘å█î╪». ┌⌐╪» ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å╪» ╪«╪º╪╡┘ç ╪¿┘ç ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º ┘ê ╪¬╪║█î█î╪▒╪º╪¬ ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¡╪│╪º╪│ ╪¿╪º╪┤╪» [171]. **╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌å┘å╪» ┌⌐╪º┘à┘╛╪º█î┘ä╪▒█î:** ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪»╪▒ ┌å┘å╪»█î┘å ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¬╪│╪¬ ┌⌐┘å█î╪» ╪¬╪º ┘é╪º╪¿┘ä█î╪¬ ╪¡┘à┘ä ╪▒╪º ╪¬╪╢┘à█î┘å ┌⌐┘å█î╪». ┌⌐╪º┘à┘╛╪º█î┘ä╪▒┘ç╪º█î ┘à╪«╪¬┘ä┘ü ┘à┘à┌⌐┘å ╪º╪│╪¬ ╪¬┘ü╪º┘ê╪¬┘ç╪º█î ╪╕╪▒█î┘ü█î ╪»╪▒ ╪▒┘ü╪¬╪º╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪º╪┤╪¬┘ç ╪¿╪º╪┤┘å╪» [172]. ## 10.4 ╪¼┘ç╪¬┘ç╪º█î ╪¬╪¡┘é█î┘é ┘ê ┌⌐╪º╪▒ ╪ó█î┘å╪»┘ç ### 10.4.1 ┘à╪¿╪º┘å█î ┘å╪╕╪▒█î **╪¬╪ú█î█î╪» ╪▒╪│┘à█î:** ╪¬╪¡┘é█î┘é╪º╪¬ ╪ó█î┘å╪»┘ç ╪¿╪º█î╪» ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¬╪ú█î█î╪» ╪▒╪│┘à█î ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪¿╪▒╪▒╪│█î ┌⌐┘å╪». ┘à╪º┘ç█î╪¬ ┘é╪╖╪╣█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪ó┘å ╪▒╪º ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ╪▒╪│┘à█î ┘à┘å╪º╪│╪¿ ┘à█î┌⌐┘å╪» ┘ê ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ╪╢┘à╪º┘å╪¬┘ç╪º█î ╪╡╪¡╪¬ ┘é┘ê█î╪¬╪▒█î ┘å╪│╪¿╪¬ ╪¿┘ç ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪│┘å╪¬█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» [173]. **┌»╪│╪¬╪▒╪┤┘ç╪º█î ╪¬╪ª┘ê╪▒█î ┘å┘ê╪╣:** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ╪│█î╪│╪¬┘à ┘å┘ê╪╣ C++ ╪│╪ñ╪º┘ä╪º╪¬ ┘å╪╕╪▒█î ╪¼╪º┘ä╪¿█î ╪»╪▒ ┘à┘ê╪▒╪» ╪▒╪º╪¿╪╖┘ç ╪¿█î┘å ╪º┘å┘ê╪º╪╣ ┘ê ┘à╪¬╪º-╪º┘å┘ê╪º╪╣ ┘à╪╖╪▒╪¡ ┘à█î┌⌐┘å╪». ╪¬╪¡┘é█î┘é ╪¿█î╪┤╪¬╪▒ ╪»╪▒ ┘à╪¿╪º┘å█î ╪¬╪ª┘ê╪▒█î ┘å┘ê╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å╪» ╪¬╪╡┘à█î┘à╪º╪¬ ╪╖╪▒╪º╪¡█î ╪▓╪¿╪º┘å ╪ó█î┘å╪»┘ç ╪▒╪º ╪ó┌»╪º┘ç ┌⌐┘å╪» [174]. **╪¬╪¡┘ä█î┘ä ┘╛█î┌å█î╪»┌»█î:** ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ┘à╪º ╪»╪º╪»┘ç┘ç╪º█î ╪¬╪¼╪▒╪¿█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪»╪î ╪¬╪¡┘ä█î┘ä ┘å╪╕╪▒█î ┘╛█î┌å█î╪»┌»█î ┘à╪¡╪º╪│╪¿╪º╪¬█î ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿█î┘å╪┤┘ç╪º█î ╪╣┘à█î┘é╪¬╪▒█î ╪¿┘ç ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ┘à┘é█î╪º╪│┘╛╪░█î╪▒█î ┘ê ┘ü╪▒╪╡╪¬┘ç╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪º╪▒╪º╪ª┘ç ╪«┘ê╪º┘ç╪» ╪»╪º╪» [175]. ### 10.4.2 ┌»╪│╪¬╪▒╪┤┘ç╪º█î ╪╣┘à┘ä█î **╪▓╪¿╪º┘å┘ç╪º█î ╪«╪º╪╡ ╪¡┘ê╪▓┘ç:** ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘╛╪º█î┘ç╪º█î ╪¿╪▒╪º█î ╪¼╪º╪│╪º╪▓█î ╪▓╪¿╪º┘å┘ç╪º█î ╪«╪º╪╡ ╪¡┘ê╪▓┘ç ╪»╪▒ C++ ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å┘å╪». ╪¬╪¡┘é█î┘é ╪»╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î DSL ┘ê ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘à█î╪¬┘ê╪º┘å╪» ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪¼╪»█î╪»█î ╪»╪▒ ╪▓┘à█î┘å┘ç┘ç╪º█î█î ┘à╪º┘å┘å╪» ┘à╪»┘ä╪│╪º╪▓█î ┘à╪º┘ä█î╪î ┘à╪¡╪º╪│╪¿╪º╪¬ ╪╣┘ä┘à█î ┘ê ╪¬┘ê╪│╪╣┘ç ╪¿╪º╪▓█î ╪¿╪º╪▓ ┌⌐┘å╪» [176]. **╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒:** ╪¬┌⌐┘å█î┌⌐┘ç╪º█î █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ┘à█î╪¬┘ê╪º┘å┘å╪» ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿┘ç ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪╣┘à╪º┘ä ╪┤┘ê┘å╪». ╪º█î┘å ╪¼┘ç╪¬ ╪¬╪¡┘é█î┘é ┘à█î╪¬┘ê╪º┘å╪» ╪¿┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î█î ┘à┘å╪¼╪▒ ╪┤┘ê╪» ┌⌐┘ç ╪«╪▒┘ê╪¼█î ╪«┘ê╪» ╪▒╪º ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┘ê ╪¿╪º╪▓╪«┘ê╪▒╪» ┌⌐╪º╪▒╪º█î█î ╪¬╪╖╪¿█î┘é ╪»┘ç┘å╪» [177]. **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿█î┘å ╪▓╪¿╪º┘å┘ç:** ┌⌐╪º╪▒ ╪ó█î┘å╪»┘ç ╪¿╪º█î╪» ╪¿╪▒╪▒╪│█î ┌⌐┘å╪» ┌⌐┘ç ┌å┌»┘ê┘å┘ç ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ┘à█î╪¬┘ê╪º┘å╪» █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿┘ç╪¬╪▒ ╪¿█î┘å C++ ┘ê ╪│╪º█î╪▒ ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▒╪º ╪¬╪│┘ç█î┘ä ┌⌐┘å╪». ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪º╪¬╪╡╪º┘ä ╪¿╪▒╪º█î ╪▓╪¿╪º┘å┘ç╪º█î█î ┘à╪º┘å┘å╪» Python╪î JavaScript ┘ê Rust ┘à█î╪¬┘ê╪º┘å╪» ╪»╪│╪¬╪▒╪│█î┘╛╪░█î╪▒█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç C++ ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪¿┘ç╪¿┘ê╪» ╪¿╪«╪┤╪» [178]. ### 10.4.3 ╪¬┘ê╪│╪╣┘ç ╪º┌⌐┘ê╪│█î╪│╪¬┘à **┌»╪│╪¬╪▒╪┤┘ç╪º█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»:** ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» C++ ╪º╪▓ ╪¬┘é┘ê█î╪¬┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪º╪¼╪▓╪º█î ┘à┘ê╪¼┘ê╪» ╪¿┘ç╪▒┘ç┘à┘å╪» ╪«┘ê╪º┘ç╪» ╪┤╪». ╪¡┘ê╪▓┘ç┘ç╪º█î█î ╪¿╪▒╪º█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ╪ó█î┘å╪»┘ç ╪┤╪º┘à┘ä: - ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ - ╪¬╪╖╪¿█î┘é ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪º┘å╪¬█î┘å╪▒ ┘ê ╪º┘ä┌»┘ê╪▒█î╪¬┘à - ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪»█î╪¿╪º┌» ┘ê ╪»╪▒┘ê┘å┘å┌»╪▒█î - ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç **╪¬┘ê╪│╪╣┘ç ╪º╪¿╪▓╪º╪▒:** ╪º┌⌐┘ê╪│█î╪│╪¬┘à ╪º╪¿╪▓╪º╪▒ ╪¡┘ê┘ä ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘ç┘å┘ê╪▓ ╪»╪▒ ╪¡╪º┘ä ╪¬┘ê╪│╪╣┘ç ╪º╪│╪¬. ╪¡┘ê╪▓┘ç┘ç╪º█î ╪º┘ê┘ä┘ê█î╪¬ ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç ╪º╪¿╪▓╪º╪▒ ╪┤╪º┘à┘ä: - ┘à╪¡█î╪╖┘ç╪º█î ╪¬┘ê╪│╪╣┘ç ╪¬╪╡┘ê█î╪▒█î ┘à╪¬╪º┌⌐┘ä╪º╪│ - ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¿╪º╪▓╪│╪º╪«╪¬╪º╪▒█î ╪ó┌»╪º┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ - ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┘ê ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ┌⌐╪º╪▒╪º█î█î - ┌å╪º╪▒┌å┘ê╪¿┘ç╪º█î ╪¬╪│╪¬ ╪│╪º╪▓┌»╪º╪▒█î ┌å┘å╪» ┌⌐╪º┘à┘╛╪º█î┘ä╪▒█î ## 10.5 ╪¬╪ú┘à┘ä╪º╪¬ ┘å┘ç╪º█î█î ┘à╪╣╪▒┘ü█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪»╪▒ C++23 ╪¿█î╪┤ ╪º╪▓ █î┌⌐ ╪¬┘é┘ê█î╪¬ ╪º┘ü╪▓╪º█î╪┤█î ╪▓╪¿╪º┘å ╪º╪│╪¬ΓÇö╪º█î╪¼╪º╪» ┌»╪│╪¬╪▒╪┤ ╪¿┘å█î╪º╪»█î ┘é╪»╪▒╪¬ ╪¿█î╪º┘å█î C++ ┘à╪¡╪│┘ê╪¿ ┘à█î╪┤┘ê╪». ╪¿╪▒╪º█î ╪º┘ê┘ä█î┘å ╪¿╪º╪▒ ╪»╪▒ ╪¬╪º╪▒█î╪« ╪▓╪¿╪º┘å╪î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪»╪│╪¬╪▒╪│█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪┤╪»┘ç ┘ê ┌⌐╪º╪▒╪º ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪º╪▒┘å╪» ┌⌐┘ç ╪»╪│╪¬┘ç┘ç╪º█î ╪¼╪»█î╪»█î ╪º╪▓ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘é╪¿┘ä╪º┘ï ╪║█î╪▒┘à┘à┌⌐┘å █î╪º ╪║█î╪▒╪╣┘à┘ä█î ╪¿┘ê╪»┘å╪». ### 10.5.1 ╪º╪▒╪▓█î╪º╪¿█î ╪¬╪║█î█î╪▒ ┘╛╪º╪▒╪º╪»╪º█î┘à **╪º╪▓ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿:** ╪¬┌⌐╪º┘à┘ä ╪º╪▓ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪╣┘ä╪º┘å█î ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç ╪¿┘ä┘ê╪║ C++ ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪▓╪¿╪º┘å█î ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ╪º╪│╪¬. ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘ç┘à┌å┘å╪º┘å ╪¼╪º█î ╪«┘ê╪» ╪▒╪º ╪«┘ê╪º┘ç╪» ╪»╪º╪┤╪¬╪î ╪¿╪º╪▓╪¬╪º╪¿ ╪▒┘ê█î┌⌐╪▒╪»█î ┘à╪│╪¬┘é█î┘à╪¬╪▒ ┘ê ┘é╪º╪¿┘ä ┘ü┘ç┘à╪¬╪▒ ╪¿╪▒╪º█î ╪¿╪│█î╪º╪▒█î ╪º╪▓ ┌⌐╪º╪▒┘ç╪º█î ╪▒╪º█î╪¼ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» [179]. **┌⌐╪» ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪»╪º╪»┘ç:** ╪¿╪º╪▓╪¬╪º╪¿ ╪│╪▒╪º┘å╪¼╪º┘à ┘╛╪º╪▒╪º╪»╪º█î┘à ΓÇ£┌⌐╪» ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪»╪º╪»┘çΓÇ¥ ╪▒╪º ╪¿┘ç ╪┤█î┘ê┘ç╪º█î ╪ó┌»╪º┘ç ╪º╪▓ ┌⌐╪º╪▒╪º█î█î ╪¿┘ç C++ ┘à█î╪ó┘ê╪▒╪». ╪º█î┘å ┘é╪º╪¿┘ä█î╪¬ ╪º┘ä┌»┘ê┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¼╪»█î╪»█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» ┌⌐┘ç ┘à╪▒╪▓┘ç╪º█î ╪¿█î┘å ┘à╪¡╪º╪│╪¿╪º╪¬ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪▒╪º ┘à╪¡┘ê ┘à█î┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪╢┘à╪º┘å╪¬┘ç╪º█î ┌⌐╪º╪▒╪º█î█î C++ ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å╪» [180]. **╪»┘à┘ê┌⌐╪▒╪º╪│█î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç:** ╪┤╪º█î╪» ┘à┘ç┘à╪¬╪▒ ╪º╪▓ ┘ç┘à┘ç╪î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▒╪º ┌⌐┘ç ┘é╪¿┘ä╪º┘ï ╪¬┘å┘ç╪º ╪¿╪▒╪º█î ┘å┘ê█î╪│┘å╪»┌»╪º┘å ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ┘ê ┘à╪¬╪«╪╡╪╡╪º┘å ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪»╪▒ ╪»╪│╪¬╪▒╪│ ╪¿┘ê╪» ╪»┘à┘ê┌⌐╪▒╪º╪¬█î┌⌐ ┘à█î┌⌐┘å┘å╪». ╪º█î┘å ╪»┘à┘ê┌⌐╪▒╪º╪│█î ┘╛╪¬╪º┘å╪│█î┘ä ╪¿╪º┘ä╪º ╪¿╪▒╪»┘å ┌⌐█î┘ü█î╪¬ ┘ê ┘é╪º╪¿┘ä█î╪¬ ┌⌐┘ä█î ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ C++ ╪▒╪º ╪»╪º╪▒╪» [181]. ### 10.5.2 ┌å╪┤┘à╪º┘å╪»╪º╪▓ ╪¿┘ä┘å╪»┘à╪»╪¬ **┘å┌»╪º┘ç ╪¿█î╪│╪¬ ╪│╪º┘ä┘ç:** ╪¿╪º ┘å┌»╪º┘ç ╪¿█î╪│╪¬ ╪│╪º┘ä ╪ó█î┘å╪»┘ç╪î ╪º┌⌐┘ê╪│█î╪│╪¬┘à C++ ╪▒╪º ┘à╪¬╪╡┘ê╪▒ ┘à█î┌⌐┘å█î┘à ┌⌐┘ç ╪»╪▒ ╪ó┘å ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ┘ç┘å╪¼╪º╪▒ ╪¿╪º╪┤╪» ╪¬╪º ╪º╪│╪¬╪½┘å╪º. ┌⌐╪» ╪ó█î┘å╪»┘ç C++ ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ╪º╪╣┘ä╪º┘å█î╪¬╪▒ ╪«┘ê╪º┘ç╪» ╪¿┘ê╪»╪î ╪¿╪º ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î█î ┌⌐┘ç ╪º┌⌐╪½╪▒ ┘å┌»╪▒╪º┘å█î┘ç╪º█î ╪▓█î╪▒╪│╪º╪«╪¬█î ╪▒╪º ┘à╪»█î╪▒█î╪¬ ┘à█î┌⌐┘å┘å╪» ┘ê ╪¿┘ç ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪º╪¼╪º╪▓┘ç ╪¬┘à╪▒┌⌐╪▓ ╪¿╪▒ ┘à┘å╪╖┘é ╪«╪º╪╡ ╪¡┘ê╪▓┘ç ┘à█î╪»┘ç┘å╪» [182]. **█î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º█î ┘å┘ê╪╕┘ç┘ê╪▒:** ┘ç┘à╪º┘å╪╖┘ê╪▒ ┌⌐┘ç ┘à╪¡╪º╪│╪¿╪º╪¬ ╪¿┘ç ╪│┘à╪¬ ╪│█î╪│╪¬┘à┘ç╪º█î ┘å╪º┘ç┘à┌»┘å╪î ╪¬┘ê╪▓█î╪╣ ╪┤╪»┘ç ┘ê ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º AI ╪º╪»╪º┘à┘ç ┘à█î╪»┘ç╪»╪î ╪¬┘ê╪º┘å╪º█î█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ┘╛┘ä ╪▓╪»┘å ╪¿█î┘å ╪│╪º╪«╪¬╪º╪▒ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘ê ╪¬╪╖╪¿█î┘é ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪º╪▒╪▓╪┤┘à┘å╪»╪¬╪▒ ╪«┘ê╪º┘ç╪» ╪┤╪». ╪º┘å╪¬╪╕╪º╪▒ ╪»╪º╪▒█î┘à ╪¿╪º╪▓╪¬╪º╪¿ ┘å┘é╪┤ ┘à┘ç┘à█î ╪»╪▒ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐╪» ╪¿╪▒╪º█î ┘╛╪º╪▒╪º╪»╪º█î┘à┘ç╪º█î ┘à╪¡╪º╪│╪¿╪º╪¬█î ╪¼╪»█î╪» ╪º█î┘ü╪º ┌⌐┘å╪» [183]. **╪¬╪¡┘ê┘ä ╪ó┘à┘ê╪▓╪┤█î:** ╪ó┘à┘ê╪▓╪┤ C++ ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ╪¬┘ê╪│╪╖ ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪¡┘ê┘ä ╪«┘ê╪º┘ç╪» █î╪º┘ü╪¬. ╪¿╪▒┘å╪º┘à┘ç┘ç╪º█î ╪»╪▒╪│█î ╪ó█î┘å╪»┘ç ┘à█î╪¬┘ê╪º┘å┘å╪» ┘à┘ü╪º┘ç█î┘à ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▒╪º ╪▓┘ê╪»╪¬╪▒ ┘ê ╪┤┘ç┘ê╪»█î╪¬╪▒ ┘à╪╣╪▒┘ü█î ┌⌐┘å┘å╪» ┘ê ╪º╪¡╪¬┘à╪º┘ä╪º┘ï ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î ╪¿╪▒╪º█î ╪¬╪«╪╡╪╡ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ╪▒╪º ┌⌐┘ê╪¬╪º┘ç ┌⌐┘å┘å╪» [184]. ### 10.5.3 ┘ü╪▒╪º╪«┘ê╪º┘å ╪╣┘à┘ä ┘à┘ê┘ü┘é█î╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪»╪▒ ╪¬╪¡┘ê┘ä ╪¬┘ê╪│╪╣┘ç C++ ╪¿┘ç ┘à╪┤╪º╪▒┌⌐╪¬ ┘ü╪╣╪º┘ä ┌⌐┘ä ╪¼╪º┘à╪╣┘ç C++ ╪¿╪│╪¬┌»█î ╪»╪º╪▒╪». ┘à╪º ╪¬╪┤┘ê█î┘é ┘à█î┌⌐┘å█î┘à: **┘╛█î╪º╪»┘ç╪│╪º╪▓╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪▒:** ╪º╪»╪º┘à┘ç ╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪¿╪º ┌⌐█î┘ü█î╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º█î ╪╣╪º┘ä█î ┘ê ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪»█î╪¿╪º┌». ╪¬╪¼╪▒╪¿┘ç ┌⌐╪º╪▒╪¿╪▒█î ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪º ╪¡╪» ╪▓█î╪º╪»█î ┘à┘ê┘ü┘é█î╪¬ ┘╛╪░█î╪▒╪┤ ╪ó┘å┘ç╪º ╪▒╪º ╪¬╪╣█î█î┘å ╪«┘ê╪º┘ç╪» ┌⌐╪▒╪». **┘å┘ê█î╪│┘å╪»┌»╪º┘å ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç:** ╪¿╪º ╪╖╪▒╪º╪¡█î┘ç╪º█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪ó╪▓┘à╪º█î╪┤ ┌⌐┘å█î╪» ┘ê ╪¬╪¼╪▒╪¿█î╪º╪¬ ╪▒╪º ╪¿╪º ╪¼╪º┘à╪╣┘ç ╪¿┘ç ╪º╪┤╪¬╪▒╪º┌⌐ ╪¿┌»╪░╪º╪▒█î╪». ┘╛╪░█î╪▒┘å╪»┌»╪º┘å ╪º┘ê┘ä█î┘ç ┘ü╪▒╪╡╪¬ ╪¬╪╣█î█î┘å ╪¿┘ç╪¬╪▒█î┘å ╪┤█î┘ê┘ç┘ç╪º█î█î ╪▒╪º ╪»╪º╪▒┘å╪» ┌⌐┘ç ╪¬┘ê╪│╪╣┘ç ╪ó█î┘å╪»┘ç ╪▒╪º ╪▒╪º┘ç┘å┘à╪º█î█î ╪«┘ê╪º┘ç╪» ┌⌐╪▒╪». **╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ╪º╪¿╪▓╪º╪▒:** ╪»╪▒ ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬┘ê╪│╪╣┘ç ╪ó┌»╪º┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ┌⌐┘å█î╪». ┘à╪▓╪º█î╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘å┘ç╪º ╪¿╪º ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à┘å╪º╪│╪¿ IDE╪î ╪│█î╪│╪¬┘à ╪│╪º╪«╪¬ ┘ê ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪¡┘ä█î┘ä ┘é╪º╪¿┘ä ╪¬╪¡┘é┘é ╪º╪│╪¬. **┘à╪▒╪¿█î╪º┘å:** ┘à┘ü╪º┘ç█î┘à ╪¿╪º╪▓╪¬╪º╪¿ ╪▒╪º ╪»╪▒ ╪¿╪▒┘å╪º┘à┘ç┘ç╪º█î ╪»╪▒╪│█î ┘ê ┘à┘ê╪º╪» ╪ó┘à┘ê╪▓╪┤█î C++ █î┌⌐┘╛╪º╪▒┌å┘ç ┌⌐┘å█î╪». ┘å╪│┘ä ╪¿╪╣╪»█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å C++ ╪¿╪º█î╪» ┘à╪¬┌⌐┘ä┘à╪º┘å ╪¿┘ê┘à█î ╪º┘ä┌»┘ê┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º╪┤┘å╪». **┘à╪¡┘é┘é╪º┘å:** ╪º╪»╪º┘à┘ç ╪¿╪▒╪▒╪│█î ┘╛█î╪º┘à╪»┘ç╪º█î ┘å╪╕╪▒█î ┘ê ╪╣┘à┘ä█î ╪¿╪º╪▓╪¬╪º╪¿ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä. ┘ü╪▒╪╡╪¬┘ç╪º█î ╪║┘å█î ╪¿╪▒╪º█î ╪¬╪¡┘é█î┘é ╪»╪▒ ╪¬┘ä╪º┘é█î ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î╪î ┘à┘ç┘å╪»╪│█î ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ┘ê ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ┘ê╪¼┘ê╪» ╪»╪º╪▒╪». ## 10.6 ╪¿█î╪º┘å█î┘ç ┘å╪¬█î╪¼┘ç┌»█î╪▒█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ┘ä╪¡╪╕┘ç ╪╣╪╖┘ü█î ╪»╪▒ ╪¬┌⌐╪º┘à┘ä ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç┘å╪». ╪¿╪º ╪º╪▒╪º╪ª┘ç ╪»╪│╪¬╪▒╪│█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪┤╪»┘ç ┘ê ┌⌐╪º╪▒╪º ╪¿┘ç ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪î ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º ┘å╪│┘ä ╪¼╪»█î╪»█î ╪º╪▓ ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å┘å╪» ┌⌐┘ç ┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î C++ ╪▒╪º ╪¡┘ü╪╕ ┘à█î┌⌐┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│ ┘ê ┌⌐█î┘ü█î╪¬ ┌⌐╪» ╪▒╪º ╪¿┘ç ╪╖┘ê╪▒ ┌å╪┤┘à┌»█î╪▒█î ╪¿┘ç╪¿┘ê╪» ┘à█î╪¿╪«╪┤╪». ╪¬╪¡┘ä█î┘ä ╪¼╪º┘à╪╣ ┘à╪º ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪╡╪▒┘ü╪º┘ï ┌⌐┘å╪¼┌⌐╪º┘ê█î┘ç╪º█î ╪ó┌⌐╪º╪»┘à█î┌⌐ ┘å█î╪│╪¬┘å╪» ╪¿┘ä┌⌐┘ç ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪╣┘à┘ä█î ┘ç╪│╪¬┘å╪» ┌⌐┘ç ┌å╪º┘ä╪┤┘ç╪º█î ┘ê╪º┘é╪╣█î ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ╪▒╪º ┘╛┘ê╪┤╪┤ ┘à█î╪»┘ç┘å╪». ╪┤┘ê╪º┘ç╪» ╪º╪▓ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ╪º┘ê┘ä█î┘ç╪î ┘à╪╣█î╪º╪▒┘ç╪º█î ┌⌐╪º╪▒╪º█î█î ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘é┘ê█î╪º┘ï ┘å╪┤╪º┘å ┘à█î╪»┘ç╪» ┌⌐┘ç ╪º█î┘å ┘ê█î┌ÿ┌»█î┘ç╪º ╪¿┘å█î╪º╪»█î ╪¿╪▒╪º█î ╪┤█î┘ê┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪»╪▒┘å C++ ╪«┘ê╪º┘ç┘å╪» ╪┤╪». ╪│┘ü╪▒ ╪º╪▓ ┘à┘ü┘ç┘ê┘à ╪¬╪º ┘╛╪░█î╪▒╪┤ ┌»╪│╪¬╪▒╪»┘ç ┘å█î╪º╪▓ ╪¿┘ç ╪¬┘ä╪º╪┤ ┘à╪»╪º┘ê┘à ┌⌐┘ä ╪¼╪º┘à╪╣┘ç C++ ╪«┘ê╪º┘ç╪» ╪»╪º╪┤╪¬. ╪¿╪º ╪º█î┘å ╪¡╪º┘ä╪î ┘à╪▓╪º█î╪º█î ╪¿╪º┘ä┘é┘ê┘çΓÇö┌⌐╪º┘ç╪┤ ┌⌐╪» boilerplate╪î ╪¿┘ç╪¿┘ê╪» ╪½╪¿╪º╪¬╪î ╪¬┘é┘ê█î╪¬ ╪¿┘ç╪▒┘ç┘ê╪▒█î ┘ê ┘╛╪º╪▒╪º╪»╪º█î┘à┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¼╪»█î╪»ΓÇö╪│╪▒┘à╪º█î┘ç┌»╪░╪º╪▒█î ┘à┘ê╪▒╪» ┘å█î╪º╪▓ ╪¿╪▒╪º█î █î┌⌐┘╛╪º╪▒┌å┌»█î ┘à┘ê┘ü┘é ╪º█î┘å ┘é╪º╪¿┘ä█î╪¬┘ç╪º ╪»╪▒ ╪º┌⌐┘ê╪│█î╪│╪¬┘à C++ ╪▒╪º ╪¬┘ê╪¼█î┘ç ┘à█î┌⌐┘å╪». ┘ç┘à╪º┘å╪╖┘ê╪▒ ┌⌐┘ç ╪»╪▒ ╪ó╪│╪¬╪º┘å┘ç ╪º█î┘å ╪╣╪╡╪▒ ╪¼╪»█î╪» ╪»╪▒ ╪¬┘ê╪│╪╣┘ç C++ ╪º█î╪│╪¬╪º╪»┘ç╪º█î┘à╪î ╪»╪▒ ┘à┘ê╪▒╪» ╪ó█î┘å╪»┘ç ╪«┘ê╪┤╪¿█î┘å ┘ç╪│╪¬█î┘à. ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘╛╪º█î┘ç╪º█î ╪¿╪▒╪º█î ╪▒┘ê█î┌⌐╪▒╪»█î ╪¿█î╪º┘å┌»╪▒╪¬╪▒╪î ┘à┘ê┘ä╪»╪¬╪▒ ┘ê ┘é╪º╪¿┘ä ┘å┌»┘ç╪»╪º╪▒█î╪¬╪▒ ╪¿┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ┘ü╪▒╪º┘ç┘à ┘à█î┌⌐┘å┘å╪» ╪»╪▒ ╪¡╪º┘ä█î ┌⌐┘ç ┘à╪┤╪«╪╡╪º╪¬ ┌⌐╪º╪▒╪º█î█î ┌⌐┘ç C++ ╪▒╪º ╪¿╪▒╪º█î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪│╪«╪¬ ╪╢╪▒┘ê╪▒█î ┘à█î┌⌐┘å╪» ╪¡┘ü╪╕ ┘à█î┌⌐┘å┘å╪». ╪ó█î┘å╪»┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪»╪▒ C++ ╪▒┘ê╪┤┘å ╪º╪│╪¬╪î ┘ê ╪¿╪º╪▓╪¬╪º╪¿█î ╪º╪│╪¬. ΓÇö ## ┘é╪»╪▒╪»╪º┘å█î┘ç╪º ┘å┘ê█î╪│┘å╪»┌»╪º┘å ┘à╪▒╪º╪¬╪¿ ┘é╪»╪▒╪»╪º┘å█î ╪«┘ê╪» ╪▒╪º ╪º╪▓ ┘à╪┤╪º╪▒┌⌐╪¬┘ç╪º█î ┌⌐┘à█î╪¬┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î C++╪î ┘╛█î╪º╪»┘ç╪│╪º╪▓╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ┘ê ╪¼╪º┘à╪╣┘ç ┌»╪│╪¬╪▒╪»┘ç C++ ┌⌐┘ç ╪¬┘ä╪º╪┤┘ç╪º█î ╪«╪│╪¬┌»█î┘å╪º┘╛╪░█î╪▒╪┤╪º┘å ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪▒╪º ╪¿┘ç ┘ê╪º┘é╪╣█î╪¬ ╪¬╪¿╪»█î┘ä ┌⌐╪▒╪»┘ç ╪º╪╣┘ä╪º┘à ┘à█î╪»╪º╪▒┘å╪». ╪¬╪┤┌⌐╪▒ ┘ê█î┌ÿ┘ç ╪º╪▓ ┘╛╪░█î╪▒┘å╪»┌»╪º┘å ╪º┘ê┘ä█î┘ç ┘ê ┘à╪¡┘é┘é╪º┘å█î ┌⌐┘ç ┌⌐╪º╪▒╪┤╪º┘å ┘╛╪º█î┘ç ╪¬╪¼╪▒╪¿█î ╪º█î┘å ╪¬╪¡┘ä█î┘ä ╪▒╪º ┘ü╪▒╪º┘ç┘à ┌⌐╪▒╪»┘ç. ╪º█î┘å ╪¬╪¡┘é█î┘é ╪¬┘ê╪│╪╖ ╪¬╪¡┘ä█î┘ä ┘å╪╕╪▒█î╪î ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪╣┘à┘ä█î ┘ê ╪º╪▒╪▓█î╪º╪¿█î ╪¼╪º┘à╪╣ ┌⌐╪º╪▒╪º█î█î ╪»╪▒ ┌å┘å╪»█î┘å ╪¡┘ê╪▓┘ç ┘ê ┘à┘ê╪▒╪» ╪º╪│╪¬┘ü╪º╪»┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪┤╪»┘ç ╪º╪│╪¬. ╪¿█î┘å╪┤┘ç╪º█î ╪º╪▒╪º╪ª┘ç ╪┤╪»┘ç ╪»╪▒ ╪º█î┘å╪¼╪º ╪¡┌⌐┘à╪¬ ╪¼┘à╪╣█î ╪¼╪º┘à╪╣┘ç C++ ╪▒╪º ┘à┘å╪╣┌⌐╪│ ┘à█î┌⌐┘å╪» ┘ê ╪¿┘ç ╪ó█î┘å╪»┘ç╪º█î ╪º┘à█î╪»┘ê╪º╪▒┌⌐┘å┘å╪»┘ç ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ╪º╪┤╪º╪▒┘ç ┘à█î┌⌐┘å╪». ΓÇö *╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê█î╪│┘å╪»┘ç:* **┘à╪¡┘à╪»╪▒╪╢╪º ╪╣┘ä█î┘╛┘ê╪▒** ╪¬╪¡┘é█î┘é╪º╪¬ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪│█î╪│╪¬┘à┘ç╪º ╪º█î┘à█î┘ä: [research.contact@domain.example] ORCID: [0000-0000-0000-0000] *┘à┌⌐╪º╪¬╪¿╪º╪¬ ┘à╪▒╪¿┘ê╪╖ ╪¿┘ç ╪º█î┘å ┘à┘é╪º┘ä┘ç ╪¿╪º█î╪» ╪¿┘ç ┘à╪¡┘à╪»╪▒╪╢╪º ╪╣┘ä█î┘╛┘ê╪▒ ╪º╪▒╪│╪º┘ä ╪┤┘ê╪».* ΓÇö *╪»╪│╪¬┘å┘ê╪┤╪¬┘ç ╪»╪▒█î╪º┘ü╪¬ ╪┤╪»┘ç: ╪ó┌»┘ê╪│╪¬ 2025╪¢ ╪¿╪º╪▓┘å┌»╪▒█î ╪┤╪»┘ç: ╪ó┌»┘ê╪│╪¬ 2025╪¢ ┘╛╪░█î╪▒┘ü╪¬┘ç ╪┤╪»┘ç: ╪ó┌»┘ê╪│╪¬ 2025* *┬⌐ 2025 ╪¬╪¡┘é█î┘é╪º╪¬ ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++. ┌⌐┘ä█î┘ç ╪¡┘é┘ê┘é ┘à╪¡┘ü┘ê╪╕ ╪º╪│╪¬.* ΓÇö *[┘à╪▒╪º╪¼╪╣ 148-184 ┘à╪╖╪º╪¿┘é ╪¿╪º ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ü┘å█î╪î ╪¬╪¡┘ä█î┘ä┘ç╪º█î ┌⌐╪º╪▒╪º█î█î╪î ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î ┘╛╪░█î╪▒╪┤╪î ┘à╪¿╪º┘å█î ┘å╪╕╪▒█î ┘ê ╪¬╪¡┘é█î┘é╪º╪¬ ┌å╪┤┘à╪º┘å╪»╪º╪▓ ╪¿┘ä┘å╪»┘à╪»╪¬ ┘ü┘ç╪▒╪│╪¬ ╪┤╪»┘ç ╪»╪▒ ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪¼╪º┘à╪╣ ┘à╪º]* # ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ┌⌐╪º┘à┘ä ╪¿╪▒╪º█î ┘à┘é╪º┘ä┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ## ┘ü┘ç╪▒╪│╪¬ ┘à╪▒╪º╪¼╪╣ (182 ┘à╪▒╪¼╪╣ █î╪º┘ü╪¬ ╪┤╪»┘ç) ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¡┘ä█î┘ä ╪│█î╪│╪¬┘à╪º╪¬█î┌⌐ ╪¬┘à╪º┘à 10 ╪¿╪«╪┤ ┘à┘é╪º┘ä┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23╪î ┘à╪▒╪º╪¼╪╣ ╪▓█î╪▒ ╪º╪│╪¬╪«╪▒╪º╪¼ ┘ê ╪»╪│╪¬┘ç╪¿┘å╪»█î ╪┤╪»┘ç╪º┘å╪». ┘ç╪▒ ╪┤┘à╪º╪▒┘ç ┘à╪▒╪¼╪╣ ┘à╪╖╪º╪¿┘é ╪¿╪º ╪º╪▒╪¼╪º╪╣╪º╪¬ █î╪º┘ü╪¬ ╪┤╪»┘ç ╪»╪▒ ╪│╪▒╪º╪│╪▒ ╪¿╪«╪┤┘ç╪º█î ┘à┘é╪º┘ä┘ç ╪º╪│╪¬. ### ┘à╪▒╪º╪¼╪╣ [1-50]: ┘à╪¿╪º┘å█î ┘ê ┘╛█î╪┤█î┘å┘ç **[1]** ┘à╪¿╪º┘å█î ╪º┘ê┘ä█î┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ C++ ┘ê ╪¬┘ê╪│╪╣┘ç ╪¬╪º╪▒█î╪«█î **[2]** ╪¬┌⌐╪º┘à┘ä ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ C++ ╪º╪▓ C++98 ╪¬╪º ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î ┘à╪»╪▒┘å **[3]** ╪│█î╪│╪¬┘à ┘é╪º┘ä╪¿ ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪│█î╪│╪¬┘à ┘à╪¡╪º╪│╪¿╪º╪¬ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┌⌐╪º┘à┘ä ╪¬┘ê╪▒█î┘å┌» **[4]** Boost.MPL (┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î) - ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪¿┘å█î╪º╪»█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ **[5]** Boost.Hana - ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ┘à╪»╪▒┘å ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬╪º╪¿╪╣█î ╪¿╪▒╪º█î C++ **[6]** ┘╛█î┌å█î╪»┌»█î ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» <type\_traits> ┘ê ╪¬╪¡┘ä█î┘ä ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î **[7]** ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪«╪º╪▒╪¼█î ┘ê ╪¬┌⌐┘å█î┌⌐┘ç╪º█î SFINAE ╪¿╪▒╪º█î ╪¬┌⌐╪▒╪º╪▒ ╪╣╪╢┘ê struct **[8]** ╪¬╪║█î█î╪▒╪º╪¬ ╪¿┘å█î╪º╪»█î ╪»╪▒ ╪¿█î╪º┘å┌»╪▒█î ┘ê ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ╪»╪▒ ╪¬┌⌐╪º┘à┘ä C++ **[9]** ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘ü╪╣┘ä█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ┘à┘é█î╪º╪│ ╪¿╪▓╪▒┌» (┘é╪│┘à╪¬ 1) **[10]** ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘ü╪╣┘ä█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┘å╪▒┘à╪º┘ü╪▓╪º╪▒ ┘à┘é█î╪º╪│ ╪¿╪▓╪▒┌» (┘é╪│┘à╪¬ 2) **[11]** ╪º┘ä┌»┘ê┘ç╪º█î ╪▒╪┤╪» ┘å┘à╪º█î█î instantiation ┘é╪º┘ä╪¿ ┘ê ┘╛█î┌å█î╪»┌»█î ┌⌐╪º┘à┘╛╪º█î┘ä O(n┬▓) **[12]** ┌⌐█î┘ü█î╪¬ ┘╛█î╪º┘à ╪«╪╖╪º█î ┘é╪º┘ä╪¿ ┘ê ┌å╪º┘ä╪┤┘ç╪º█î ╪«╪▒┘ê╪¼█î ┘à╪¿┘ç┘à ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ **[13]** ┘é╪º╪¿┘ä█î╪¬┘ç╪º█î ╪»╪▒┘ê┘å┘å┌»╪▒█î ┘à╪¡╪»┘ê╪» ┘ê ╪┤┌⌐┘å┘å╪»┌»█î ╪▒╪º┘ç╪¡┘ä┘ç╪º█î SFINAE **[14]** ┌å╪º┘ä╪┤┘ç╪º█î ┘é╪º╪¿┘ä█î╪¬ ┘å┌»┘ç╪»╪º╪▒█î ╪»╪▒ ┌⌐╪» ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪¿╪▒╪º█î ┘à╪¡█î╪╖┘ç╪º█î ╪│╪º╪▓┘à╪º┘å█î **[15]** ╪¬┘ê╪▒┘à ╪¿╪º█î┘å╪▒█î ╪º╪▓ instantiation ╪¿█î╪┤ ╪º╪▓ ╪¡╪» ┘é╪º┘ä╪¿ ┘ê inlining ╪¬┘ç╪º╪¼┘à█î **[16]** ┘à╪╣╪▒┘ü█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ C++23 ╪º╪▒╪º╪ª┘ç ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪»╪▒╪¼┘ç █î┌⌐ ╪▓╪¿╪º┘å ╪¿╪▒╪º█î ╪»╪▒┘ê┘å┘å┌»╪▒█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[17]** ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¡┘ü╪╕ ╪╡┘ü╪▒ overhead ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪»╪▒ ╪¡█î┘å ┘ü╪╣╪º┘ä╪│╪º╪▓█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[18]** ╪»╪│╪¬╪▒╪│█î ┘à╪│╪¬┘é█î┘à ╪¿┘ç ┘å┘à╪º█î╪┤ ╪»╪º╪«┘ä█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ **[19]** ╪º╪│┘å╪º╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ISO C++23 ┘ê ╪¬╪¡┘ä█î┘ä ┘à┘é╪º┘ä╪º╪¬ ┘╛█î╪┤┘å┘ç╪º╪»█î ┘à╪▒╪¬╪¿╪╖ (P0194) **[20]** ╪º╪│┘å╪º╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ISO C++23 ┘ê ╪¬╪¡┘ä█î┘ä ┘à┘é╪º┘ä╪º╪¬ ┘╛█î╪┤┘å┘ç╪º╪»█î ┘à╪▒╪¬╪¿╪╖ (P0385) **[21]** ╪º╪│┘å╪º╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ISO C++23 ┘ê ╪¬╪¡┘ä█î┘ä ┘à┘é╪º┘ä╪º╪¬ ┘╛█î╪┤┘å┘ç╪º╪»█î ┘à╪▒╪¬╪¿╪╖ (P0707) **[22]** ┘é╪º┘ä╪¿┘ç╪º█î variadic C++11 ╪¬╪¡┘ê┘ä ╪»╪▒ ┌å╪┤┘à╪º┘å╪»╪º╪▓ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î **[23]** Perfect forwarding ╪¿╪º ╪º╪▒╪¼╪º╪╣╪º╪¬ rvalue ┘ê ╪º╪▒╪¼╪º╪╣╪º╪¬ ╪¼┘ç╪º┘å█î (T&&) **[24]** ╪¿┘ç╪¿┘ê╪»┘ç╪º█î SFINAE ╪¿╪º std::enable\_if ┘ê type traits **[25]** ╪¬┘ê╪º╪¿╪╣ constexpr ╪¿╪▒╪º█î ╪º╪▒╪▓█î╪º╪¿█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘╛┘ä ╪▓╪»┘å ╪┤┌⌐╪º┘ü ╪▓┘à╪º┘å ╪º╪¼╪▒╪º/╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[26]** ╪¬╪╡╪¡█î╪¡╪º╪¬ ┘à╪»┘ä ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î C++14 ┘ê ╪¿┘ç╪¿┘ê╪»┘ç╪º█î ╪«┘ê╪º┘å╪º█î█î **[27]** ┘ä╪º┘à╪¿╪»╪º┘ç╪º█î ╪╣┘à┘ê┘à█î ╪¿╪º ┘╛╪º╪▒╪º┘à╪¬╪▒┘ç╪º█î auto ┘ü╪╣╪º┘ä╪│╪º╪▓█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬╪º╪¿╪╣█î **[28]** constexpr ┌»╪│╪¬╪▒╪»┘ç █î╪º┘ü╪¬┘ç ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪ó╪▓╪º╪» ╪┤╪»┘ç ╪¿╪▒╪º█î ┘à╪¡╪º╪│╪¿╪º╪¬ ┘╛█î┌å█î╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[29]** concept ┘ç╪º█î C++20 ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘à┘ç┘à╪¬╪▒█î┘å ┘╛█î╪┤╪▒┘ü╪¬ ╪º╪▓ ╪▓┘à╪º┘å ┘é╪º┘ä╪¿┘ç╪º█î variadic **[30]** ╪¿┘ç╪¿┘ê╪» concept ┘ç╪º ╪»╪▒ ╪¬╪┤╪«█î╪╡ ╪«╪╖╪º█î ┘é╪º┘ä╪¿ ┘ê ┘å┘é╪╢ ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º **[31]** ╪¬┘é┘ê█î╪¬ concept ┘ç╪º ╪»╪▒ ╪¡┘ä overload ╪¿╪º ┘é┘ê╪º┘å█î┘å ╪¬┘é╪»┘à ┘ê╪º╪╢╪¡ **[32]** ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪¿┘å█î╪º╪»█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ┘é╪º┘ä╪¿ **[33]** ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘╛█î┌å█î╪»┌»█î ┌⌐╪º┘à┘╛╪º█î┘ä ┘å╪┤╪º┘å╪»┘ç┘å╪»┘ç 60-80% ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ ┌⌐╪»╪¿█î╪│┘ç╪º█î ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ **[34]** ╪¿╪º╪▒ ╪┤┘å╪º╪«╪¬█î ┘ê ┘à┘ê╪º┘å╪╣ ╪¬╪«╪╡╪╡ ╪»╪▒ ╪¬█î┘à┘ç╪º█î ╪¿╪▓╪▒┌» **[35]** ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪│╪º█î╪▒ ╪▓╪¿╪º┘å┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪ó┌»╪º┘ç┌⌐┘å┘å╪»┘ç ╪╖╪▒╪º╪¡█î C++ **[36]** ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ Java ┘à╪╣╪▒┘ü█î ╪┤╪»┘ç ╪»╪▒ Java 1.1 ╪¿╪º ╪»╪▒┘ê┘å┘å┌»╪▒█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º **[37]** overhead ┌⌐╪º╪▒╪º█î█î ╪¿╪º╪▓╪¬╪º╪¿ Java: 10-100x ┌⌐┘å╪»╪¬╪▒ ╪º╪▓ ┘ü╪▒╪º╪«┘ê╪º┘å█î ┘à╪│╪¬┘é█î┘à **[38]** ┘à╪»█î╪▒█î╪¬ ╪º┘à┘å█î╪¬ ╪¿╪º╪▓╪¬╪º╪¿ Java ┘ê ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪»╪│╪¬╪▒╪│█î ╪║█î╪▒┘à╪¼╪º╪▓ **[39]** ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ C# ╪│╪º╪«╪¬┘ç ╪¿╪▒ Java ╪¿╪º ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[40]** ╪»╪▒╪«╪¬┘ç╪º█î ╪¿█î╪º┘å C# ╪¿╪▒╪º█î ┘å┘à╪º█î╪┤ ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä (Entity Framework) **[41]** ╪¬┘ê┘ä█î╪»┌⌐┘å┘å╪»┘ç┘ç╪º█î ┘à┘å╪¿╪╣ C# ╪º╪▒╪º╪ª┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘à╪º┘å┘å╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 **[42]** ┘à╪º┌⌐╪▒┘ê┘ç╪º█î ╪▒┘ê█î┘ç╪º█î Rust ╪╣┘à┘ä ╪¿╪▒ ╪»╪▒╪«╪¬ ┘å╪¡┘ê ╪º┘å╪¬╪▓╪º╪╣█î (AST) ╪╖█î ┌⌐╪º┘à┘╛╪º█î┘ä **[43]** ╪º╪¼╪▒╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ┘à╪º┌⌐╪▒┘ê┘ç╪º█î Rust ╪¿╪º ╪º┘å╪¬╪▓╪º╪╣╪º╪¬ ╪¿╪»┘ê┘å ┘ç╪▓█î┘å┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º **[44]** ╪╢┘à╪º┘å╪¬┘ç╪º█î ╪¿┘ç╪»╪º╪┤╪¬ ╪│█î╪│╪¬┘à ┘à╪º┌⌐╪▒┘ê█î Rust ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪º╪│█î╪▒ ┘å╪º┘à **[45]** ╪▓╪¿╪º┘å ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î D ┘╛█î╪┤┌»╪º┘à ┘à┘ü╪º┘ç█î┘à ╪¬╪ú╪½█î╪▒┌»╪░╪º╪▒ ╪¿╪▒ ╪╖╪▒╪º╪¡█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 **[46]** ╪»╪▒┘ê┘å┘å┌»╪▒█î ╪º╪│╪¬╪º╪¬█î┌⌐ D ╪º╪▒╪º╪ª┘ç ╪º╪╖┘ä╪º╪╣╪º╪¬ ┘å┘ê╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪»┘ê┘å overhead ╪▓┘à╪º┘å ╪º╪¼╪▒╪º **[47]** ╪º╪¼╪▒╪º█î ╪¬╪º╪¿╪╣ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä D (CTFE) ┘ê mixin ┘ç╪º█î ╪▒╪┤╪¬┘ç ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» **[48]** ╪¬┘ä╪º╪┤┘ç╪º█î ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪»╪▒ ╪╖┘ê┘ä ╪│╪º┘ä┘ç╪º ┘ê ┘╛█î╪┤┘å┘ç╪º╪»┘ç╪º█î ┘à╪¬╪╣╪»╪» **[49]** P0194 ┌⌐╪º╪▒ ╪¿┘å█î╪º╪»█î ╪¬┘ê╪│╪╖ Mat├║┼í Chochl├¡k╪î Axel Naumann ┘ê David Sankel **[50]** ┘à┘å╪╖┘é ╪¬┘ü╪╡█î┘ä█î ╪¿╪▒╪º█î ╪¬╪╡┘à█î┘à╪º╪¬ ╪╖╪▒╪º╪¡█î ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ ### ┘à╪▒╪º╪¼╪╣ [51-100]: ┌å╪º╪▒┌å┘ê╪¿ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ╪╖╪▒╪º╪¡█î **[51]** ┘╛█î╪┤┘å┘ç╪º╪» ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î Herb Sutter ┘à╪╣╪▒┘ü█î ╪¬┘ê┘ä█î╪» ┌⌐┘ä╪º╪│ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[52]** ╪¬╪╡╪¡█î╪¡ ┘ü╪▒╪ó█î┘å╪» ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»╪│╪º╪▓█î ╪¿╪▒ ╪º╪│╪º╪│ ╪¬╪¼╪▒╪¿┘ç ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ╪¿╪º╪▓╪«┘ê╪▒╪» **[53]** ╪¬╪╣╪º╪»┘ä┘ç╪º ╪¿█î┘å ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘ê ┘╛┘ê█î╪º **[54]** ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ┘à╪«╪¬┘ä┘ü █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à┘ä╪º╪¡╪╕╪º╪¬ ╪╖╪▒╪º╪¡█î **[55]** ╪¬╪ú╪½█î╪▒╪º╪¬ ╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒ ╪º┌⌐┘ê╪│█î╪│╪¬┘à┘ç╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î **[56]** ╪¡┘ê╪▓┘ç┘ç╪º█î█î ┌⌐┘ç ╪¬╪¡┘é█î┘é ╪¿╪º╪▓╪¬╪º╪¿ C++23 ┘à█î╪¬┘ê╪º┘å╪» ┘à╪┤╪º╪▒┌⌐╪¬┘ç╪º█î ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ┌⌐┘å╪» **[57]** ┘╛╪▒┘ê╪¬┌⌐┘ä meta-object ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒╪º█î ╪»╪│╪¬╪▒╪│█î ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[58]** ┘ê╪¼┘ê╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä meta-object ┘ç╪º ╪¬╪╢┘à█î┘å ╪╡┘ü╪▒ overhead ╪▓┘à╪º┘å ╪º╪¼╪▒╪º **[59]** ╪»┘å╪¿╪º┘ä┘ç┘ç╪º█î meta-object ╪¿╪▒╪º█î ┘╛╪▒╪»╪º╪▓╪┤ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪º ┌»╪│╪¬╪▒╪┤ ╪¿╪│╪¬┘ç ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ **[60]** ╪¬┌⌐╪▒╪º╪▒ ┌⌐╪º╪▒╪º ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪»┘ê┘å ┌⌐╪º┘å╪¬█î┘å╪▒┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º █î╪º ╪¿╪º╪▓┌»╪┤╪¬ ┘╛█î┌å█î╪»┘ç ┘é╪º┘ä╪¿ **[61]** ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪▒╪▒╪│█î ┘é┘ê█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪»╪▒ API ╪¿╪º╪▓╪¬╪º╪¿ **[62]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿█î╪»╪▒╪▓ ╪¿╪º ┘ê█î┌ÿ┌»█î┘ç╪º█î ┘à┘ê╪¼┘ê╪» C++╪î ┘é╪º┘ä╪¿┘ç╪º ┘ê concept ┘ç╪º **[63]** ╪╖╪▒╪º╪¡█î ╪¿╪»┘ê┘å ┘ç╪▓█î┘å┘ç ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ╪º╪▒╪▓█î╪º╪¿█î constexpr ┘ê ╪¿╪│╪¬┘ç┘ç╪º█î ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ **[64]** ╪º╪│╪¬┘å╪¬╪º╪¼ ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘é╪º┘ä╪¿ ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¡╪░┘ü ╪│╪º╪«╪¬╪º╪▒┘ç╪º█î ┘╛█î┌å█î╪»┘ç SFINAE **[65]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪º ┘é╪º┘ä╪¿┘ç╪º█î variadic ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ┘é╪»╪▒╪¬┘à┘å╪» **[66]** ┌⌐╪º┘ç╪┤ ┌å╪┤┘à┌»█î╪▒ ╪»╪▒ ╪¬╪«╪╡╪╡┘ç╪º█î ┘é╪º┘ä╪¿ ┘à┘ê╪▒╪» ┘å█î╪º╪▓ **[67]** ╪╣┘à┘ä┌»╪▒ reflexpr ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘å┘é╪╖┘ç ┘ê╪▒┘ê╪» ╪º╪╡┘ä█î ╪¿┘ç ╪│█î╪│╪¬┘à ╪¿╪º╪▓╪¬╪º╪¿ **[68]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¬╪╢┘à█î┘å ╪º┘ç╪»╪º┘ü ╪¿╪º╪▓╪¬╪º╪¿ ┘à╪╣╪¬╪¿╪▒ **[69]** ╪»╪│╪¬┘ç┘ç╪º█î meta-object ╪¿╪º ╪▒╪º╪¿╪╖┘ç╪º█î ┌⌐┘ê╪ª╪▒█î ╪«╪º╪╡ **[70]** ╪▓┘à█î┘å┘ç┘ç╪º█î ╪º╪▒╪▓█î╪º╪¿█î constexpr ╪¬╪╢┘à█î┘å ╪º╪¼╪▒╪º█î ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[71]** ┌⌐╪º┘à┘╛╪º█î┘ä ╪┤╪▒╪╖█î ┘╛█î┌å█î╪»┘ç ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘ê╪º┘é╪╣█î ┘å┘ê╪╣ **[72]** ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┌⌐┘ä╪º╪│┘ç╪º█î adapter ┘ê proxy **[73]** ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ╪▒╪º█î╪¼ **[74]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪«╪╖ ┘ä┘ê┘ä┘ç ┌⌐╪º┘à┘╛╪º█î┘ä ╪¡┘ü╪╕ ╪º╪╡┘ê┘ä ┌⌐╪º┘à┘╛╪º█î┘ä ╪¼╪»╪º┌»╪º┘å┘ç **[75]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» ┘ê ╪¬┘ê╪│╪╣┘ç **[76]** ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¿┘ç╪▒┘ç╪¿╪▒╪»╪º╪▒█î ╪º╪▓ ╪º╪╖┘ä╪º╪╣╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç **[77]** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪º┘ê╪¼ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¬┘ê┘ä█î╪»█î ┘ê ╪¬╪¡┘é█î┘é ╪│┘å╪¬╪▓ ┌⌐╪» ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä **[78]** ╪¿█î┘å╪┤ ╪¿┘å█î╪º╪»█î ╪»╪▒ ┘à┘ê╪▒╪» ┌⌐╪» boilerplate ╪¬┌⌐╪▒╪º╪▒█î ┘╛█î╪▒┘ê█î ╪º╪▓ ╪º┘ä┌»┘ê┘ç╪º█î ┘é╪º╪¿┘ä ┘╛█î╪┤╪¿█î┘å█î **[79]** ┘à╪┤╪«╪╡╪º╪¬ ╪º╪╣┘ä╪º┘å█î ╪▒┘ü╪¬╪º╪▒ ┘à╪╖┘ä┘ê╪¿ ╪¿╪º ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î **[80]** ╪º╪╡┘ê┘ä ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ C++23 **[81]** ╪▒╪º╪¿╪╖┘ç ┘ç┘à╪▓█î╪│╪¬ ╪¿█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘ê ╪▓█î╪▒╪│╪º╪«╪¬ ╪¿╪º╪▓╪¬╪º╪¿ **[82]** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¬╪╣╪▒█î┘ü ╪┤╪»┘ç ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪¬┘ê╪º╪¿╪╣ constexpr ╪╣┘à┘ä ╪¿╪▒ ╪ó╪¿╪¼┌⌐╪¬┘ç╪º█î std::meta::info **[83]** ┘╛╪░█î╪▒╪┤ ┘╛╪º╪▒╪º┘à╪¬╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒╪º█î ╪│┘ü╪º╪▒╪┤█î╪│╪º╪▓█î ╪▒┘ü╪¬╪º╪▒ **[84]** ┘à┘å╪╖┘é ╪┤╪▒╪╖█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪▒ ╪º╪│╪º╪│ ┘à╪┤╪«╪╡╪º╪¬ ┘å┘ê╪╣ **[85]** ╪▒╪º╪¿╪╖ std::meta::compiler ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘à┌⌐╪º┘å█î╪▓┘à ╪º╪╡┘ä█î ╪¬┘ê┘ä█î╪» ┌⌐╪» **[86]** ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘é╪º┘ä╪¿ ╪»╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪º█î┘à┘å ┘å┘ê╪╣ **[87]** ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î┘ç╪º█î ┘╛█î┌å█î╪»┘ç ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪¿╪▒╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘╛█î┌å█î╪»┘ç **[88]** ╪¬┘ê┘ä█î╪» ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪▒╪º╪¿╪╖ ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪▒ ╪º╪│╪º╪│ ╪º┘ä┌»┘ê┘ç╪º **[89]** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘à╪»█î╪▒█î╪¬ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ┘╛█î┌å█î╪»┘ç **[90]** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ┘à╪¬┘à╪▒┌⌐╪▓ ╪¿╪▒ ┘╛╪º█î┌»╪º┘ç ╪»╪º╪»┘ç ╪¿╪º ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘╛█î┌å█î╪»┘ç **[91]** ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘╛█î┌å█î╪»┘ç **[92]** ╪¬╪▒┌⌐█î╪¿ ┘é╪º╪¿┘ä ┘╛█î╪┤╪¿█î┘å█î ┌å┘å╪»█î┘å ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪╣┘à╪º┘ä ╪┤╪»┘ç ╪¿┘ç ┘ç┘à╪º┘å ┘å┘ê╪╣ **[93]** ╪º╪▒╪¬╪¿╪º╪╖ ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪º╪▓ ╪╖╪▒█î┘é ┘à╪¬╪º╪»█î╪¬╪º█î ┘à╪┤╪¬╪▒┌⌐ ┘ê ┘é╪▒╪º╪▒╪»╪º╪»┘ç╪º **[94]** ┘à╪»█î╪▒█î╪¬ ╪╡╪▒█î╪¡ ┘ê╪º╪¿╪│╪¬┌»█î ╪¿╪▒╪º█î ╪¬╪╣╪º┘à┘ä╪º╪¬ ┘╛█î┌å█î╪»┘ç ┘à╪¬╪º┌⌐┘ä╪º╪│ **[95]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌»╪│╪¬╪▒╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒╪º█î ╪¬╪┤╪«█î╪╡ ╪▓┘ê╪»┘ç┘å┌»╪º┘à ╪«╪╖╪º **[96]** ┘╛█î╪º┘à┘ç╪º█î ╪¬╪┤╪«█î╪╡█î ┘à┘ü█î╪» ╪»╪▒ ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î ╪«┘ê╪¿ ╪╖╪▒╪º╪¡█î ╪┤╪»┘ç **[97]** ╪▒┘ê╪┤╪┤┘å╪º╪│█î ╪¬╪¼╪▒╪¿█î ╪»┘é█î┘é ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î **[98]** ┘à╪¼┘à┘ê╪╣┘ç ╪¬╪│╪¬ ╪¼╪º┘à╪╣ ╪¿╪▒╪º█î ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌⌐╪º╪▒╪º█î█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ **[99]** ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘à╪╣█î╪º╪▒ ╪¡█î╪º╪¬█î ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç ┘à┘é█î╪º╪│ ╪¿╪▓╪▒┌» **[100]** ┘ç┘à╪¿╪│╪¬┌»█î ┌⌐╪º┘ç╪┤ ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¡╪º┘ü╪╕┘ç ╪¿╪º ┌⌐╪º┘ç╪┤ ╪╣┘à┘é instantiation ┘é╪º┘ä╪¿ ### ┘à╪▒╪º╪¼╪╣ [101-150]: ┌⌐╪º╪▒╪º█î█î ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î **[101]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º┘ä╪▓╪º┘à ╪╡┘ü╪▒ overhead ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪º╪▓ ╪╖╪▒█î┘é ┘à╪╣█î╪º╪▒╪│┘å╪¼█î ╪¼╪º┘à╪╣ **[102]** ╪¬╪¡┘ä█î┘ä ╪¬┘ü╪╡█î┘ä█î assembly ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ╪º╪»╪╣╪º┘ç╪º█î ╪╡┘ü╪▒-overhead **[103]** ╪¬╪¡┘ä█î┘ä ┘╛█î╪º┘à╪»┘ç╪º█î ┌⌐╪º╪▒╪º█î█î ┌⌐╪┤ ╪¬┘ê┘ä█î╪» ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ **[104]** ╪¬╪¡┘ä█î┘ä ╪¬╪ú╪½█î╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪¿╪▒╪º█î ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪º╪│╪¬┘é╪▒╪º╪▒ **[105]** ┘à╪┤╪º╪▒┌⌐╪¬ ╪¬┘ê╪▒┘à instantiation ┘é╪º┘ä╪¿ ╪»╪▒ ╪º┘å╪»╪º╪▓┘ç ╪¿╪º█î┘å╪▒█î ╪»╪▒ ┌⌐╪»╪¿█î╪│┘ç╪º█î ╪│┘å┌»█î┘å ┘é╪º┘ä╪¿ **[106]** ╪¬╪ú┌⌐█î╪» ╪»┘é╪¬ ╪ó┘à╪º╪▒█î ┘ê ╪¬┌⌐╪▒╪º╪▒┘╛╪░█î╪▒█î ╪»╪▒ ╪▒┘ê╪┤╪┤┘å╪º╪│█î ┘à╪╣█î╪º╪▒╪│┘å╪¼█î **[107]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ╪¿╪▒╪º█î ╪¬╪╣┘à█î┘à┘╛╪░█î╪▒█î ┘å╪¬╪º█î╪¼ **[108]** ┘à┘é╪º█î╪│┘ç ╪¼╪º┘à╪╣ ╪»╪▒ ╪º╪¿╪╣╪º╪» ┘à╪¬╪╣╪»╪» **[109]** ╪º┘å╪»╪º╪▓┘ç┌»█î╪▒█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç ╪º╪▓ ╪╖╪▒█î┘é ╪ó╪▓┘à╪º█î╪┤┘ç╪º█î ┌⌐┘å╪¬╪▒┘ä ╪┤╪»┘ç **[110]** ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¿┘ç ╪╣┘å┘ê╪º┘å ┌⌐╪º╪▒ ╪▒╪º█î╪¼ ╪¬┌⌐╪▒╪º╪▒█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪¿╪º ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪│┘å╪¬█î **[111]** ╪¬┘ê╪│╪╣┘ç ┌å╪º╪▒┌å┘ê╪¿ ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¼╪º┘à╪╣ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 **[112]** ╪º╪▒╪▓█î╪º╪¿█î ┌⌐╪º╪▒╪º█î█î ┘à┘é╪º█î╪│┘ç ┌å╪º╪▒┌å┘ê╪¿ ╪¿╪º ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à┘ê╪¼┘ê╪» **[113]** ┘å┌»╪º╪┤╪¬ ╪ó╪¿╪¼┌⌐╪¬-╪▒╪º╪¿╪╖┘ç╪º█î (ORM) ╪¡┘ê╪▓┘ç ┘╛█î┌å█î╪»┘ç ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪▒╪▓╪┤ ┘é╪º╪¿┘ä ╪¬┘ê╪¼┘ç█î ╪º╪▒╪º╪ª┘ç ┘à█î╪»┘ç╪» **[114]** ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪º╪╣┘ä╪º┘å█î ╪¬┘ê╪│╪╣┘ç GUI ┘à╪»╪▒┘å ┘ü╪╣╪º┘ä ╪┤╪»┘ç ╪¬┘ê╪│╪╖ ╪¿╪º╪▓╪¬╪º╪¿ C++23 **[115]** ╪¡┘ê╪▓┘ç ╪¬╪│╪¬ ┌⌐┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪¬╪│╪¬ ╪▒╪º ╪º┘à┌⌐╪º┘å┘╛╪░█î╪▒ ┘à█î┌⌐┘å╪» **[116]** ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪º┘ä┌»┘ê┘ç╪º█î ╪╖╪▒╪º╪¡█î ┘╛█î┌å█î╪»┘ç ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ **[117]** █î┌⌐┘╛╪º╪▒┌å┌»█î concept ┘ç╪º█î C++20 ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒╪º█î ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪º█î┘à┘å ┘å┘ê╪╣ **[118]** ╪¬╪╣╪º╪▒█î┘ü ┘╛█î┌å█î╪»┘ç concept ╪¿╪▒╪▒╪│█î ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ ╪¬╪º ╪▒╪º╪¿╪╖┘ç╪º **[119]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┘╛█î┌å█î╪»┘ç ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪º╪▓ ╪╖╪▒█î┘é concept ┘ç╪º ┘ê ╪¿╪º╪▓╪¬╪º╪¿ ╪¬╪▒┌⌐█î╪¿█î **[120]** coroutine ┘ç╪º█î C++20 ╪¬╪▒┌⌐█î╪¿ ╪┤╪»┘ç ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ C++23 ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ API ┘å╪º┘ç┘à╪▓┘à╪º┘å **[121]** ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ╪¼╪▒█î╪º┘å┘ç╪º█î ╪▒┘ê█î╪»╪º╪» ┘ê╪º┌⌐┘å╪┤█î ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ **[122]** ┘à╪º┌ÿ┘ê┘ä┘ç╪º█î C++20 ┘å█î╪º╪▓ ╪¿┘ç ┘à┘ä╪º╪¡╪╕╪º╪¬ ┘ê█î┌ÿ┘ç ╪¿╪▒╪º█î ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¿╪º╪▓╪¬╪º╪¿ **[123]** ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┌⌐╪º╪▒ ╪╡╪¡█î╪¡ ╪»╪▒ ┘à╪▒╪▓┘ç╪º█î ┘à╪º┌ÿ┘ê┘ä **[124]** ╪¬┘é┘ê█î╪¬ range ┘ç╪º█î C++20 ╪¿╪º ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ┘╛╪▒╪»╪º╪▓╪┤ ╪«┘ê╪»┌⌐╪º╪▒ ╪»╪º╪»┘ç **[125]** ╪¬┘ê┘ä█î╪» adapter range ╪│┘ü╪º╪▒╪┤█î ╪¿╪▒ ╪º╪│╪º╪│ ╪│╪º╪«╪¬╪º╪▒ ┘å┘ê╪╣ **[126]** ╪¬┘é┘ê█î╪¬ ┌⌐╪º┘å╪¬█î┘å╪▒ ┌⌐╪¬╪º╪¿╪«╪º┘å┘ç ╪º╪│╪¬╪º┘å╪»╪º╪▒╪» ╪¿╪º ╪╣┘à┘ä┌⌐╪▒╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ **[127]** ┌å╪º┘ä╪┤┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ╪»╪▒ frontend ┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ ╪¿╪▒╪º█î ╪¿╪º╪▓╪¬╪º╪¿ C++23 **[128]** ┘╛█î┌å█î╪»┌»█î┘ç╪º█î instantiation ┘é╪º┘ä╪¿ ╪¿╪º meta-object ┘ç╪º ╪╖█î instantiation **[129]** ┌å╪º┘ä╪┤┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î backend ╪¿╪▒╪º█î ╪¬┘ê┘ä█î╪» ┌⌐╪» ┌⌐╪º╪▒╪º ╪º╪▓ ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ **[130]** ┌å╪º┘ä╪┤┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪│█î╪│╪¬┘à┘ç╪º█î ┌⌐╪º┘à┘╛╪º█î┘ä ╪º┘ü╪▓╪º█î╪┤█î **[131]** ┌å╪º┘ä╪┤┘ç╪º█î ╪»█î╪¿╪º┌» ╪¿╪▒╪º█î ╪º╪│╪¬┘ü╪º╪»┘ç ┌»╪│╪¬╪▒╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│ **[132]** ╪º┘ä╪▓╪º┘à╪º╪¬ ╪¬┘é┘ê█î╪¬ IDE ┘à╪»╪▒┘å ╪¿╪▒╪º█î ┘╛╪┤╪¬█î╪¿╪º┘å█î ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ **[133]** ╪»╪│╪¬┘ç┘ç╪º█î ╪¼╪»█î╪» ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º█î ┘╛█î┌å█î╪»┘ç ┘à╪╣╪▒┘ü█î ╪┤╪»┘ç ╪¬┘ê╪│╪╖ ╪¿╪º╪▓╪¬╪º╪¿ **[134]** ╪º┘ä╪▓╪º┘à╪º╪¬ ┌»╪▓╪º╪▒╪┤ ╪«╪╖╪º█î ╪¬╪«╪╡╪╡█î ╪¿╪▒╪º█î ╪«╪╖╪º┘ç╪º█î ┘à╪¬╪º┌⌐┘ä╪º╪│ **[135]** ╪º╪╢╪º┘ü┘ç ╪┤╪»┘å ┘╛█î┌å█î╪»┌»█î ┘à┘ü┘ç┘ê┘à█î ╪¿┘ç C++ ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º **[136]** ┌å╪º┘ä╪┤┘ç╪º█î ╪│╪º╪▓┘à╪º┘å█î ╪»╪▒ ┘à┘ç╪º╪¼╪▒╪¬ ┌⌐╪»╪¿█î╪│┘ç╪º█î ┘à┘ê╪¼┘ê╪» ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ **[137]** ┌å╪º┘ä╪┤┘ç╪º█î ╪½╪¿╪º╪¬ ╪▒╪º╪¿╪╖ ╪¿╪º█î┘å╪▒█î ┌⌐╪º╪▒╪¿╪▒╪» (ABI) ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º **[138]** ┌å╪º┘ä╪┤┘ç╪º█î ╪▒┘ü╪¬╪º╪▒ ╪½╪º╪¿╪¬ ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º█î ┘à╪«╪¬┘ä┘ü ┌⌐╪º┘à┘╛╪º█î┘ä╪▒ **[139]** ╪╣┘ä╪º┘é┘ç ┘ü╪▓╪º█î┘å╪»┘ç ╪¿┘ç ┌»╪│╪¬╪▒╪┤ ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¿┘ç ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º **[140]** ┘à┌⌐╪º┘å█î╪▓┘à┘ç╪º█î ╪¬╪▒┌⌐█î╪¿ ┘à╪¬╪º┌⌐┘ä╪º╪│ ┘╛█î┌å█î╪»┘ç╪¬╪▒ ╪»╪▒ ╪º╪│╪¬╪º┘å╪»╪º╪▒╪»┘ç╪º█î ╪ó█î┘å╪»┘ç **[141]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿┘ç╪¬╪▒ ╪¿█î┘å ╪¿╪º╪▓╪¬╪º╪¿╪î ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º ┘ê ╪│█î╪│╪¬┘à ┘à╪º┌ÿ┘ê┘ä┘ç╪º **[142]** ┘ü╪▒╪╡╪¬┘ç╪º█î █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ╪¿╪º ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ **[143]** ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«╪º╪╡ ┘╛┘ä╪¬┘ü╪▒┘à ┘ê ╪│╪º╪▓┌»╪º╪▒█î ┌å┘å╪»┘╛┘ä╪¬┘ü╪▒┘à┘ç ╪º╪▓ ╪╖╪▒█î┘é ╪¿╪º╪▓╪¬╪º╪¿ **[144]** ╪¬┌⌐╪º┘à┘ä ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘à╪¡╪»┘ê╪»█î╪¬┘ç╪º█î ╪│█î╪│╪¬┘à ╪¿┘ä╪º╪»╪▒┘å┌» ┘ê ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç **[145]** ┘╛╪┤╪¬█î╪¿╪º┘å█î ┘╛█î┌å█î╪»┘ç ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪»█î╪¿╪º┌» ╪ó█î┘å╪»┘ç ╪¿╪▒╪º█î ┌⌐╪» ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ **[146]** ╪¬┌⌐╪º┘à┘ä IDE ╪¿╪▒╪º█î ┘╛╪┤╪¬█î╪¿╪º┘å█î ╪¼╪º┘à╪╣ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º **[147]** ╪º╪¿╪▓╪º╪▒┘ç╪º█î ╪¬╪«╪╡╪╡█î ╪¿╪▒╪º█î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│ **[148]** ╪º┘å╪¬╪▓╪º╪╣ std::meta::info ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘╛╪º█î┘ç ┘à╪¡┌⌐┘à ╪¡┘ü╪╕ ╪º╪╡┘ä ╪╡┘ü╪▒-overhead **[149]** ╪«┘ê╪»┌⌐╪º╪▒╪│╪º╪▓█î ┘à╪¬╪º┌⌐┘ä╪º╪│ 80% ┌⌐╪» boilerplate ╪»╪▒ ╪│┘å╪º╪▒█î┘ê┘ç╪º█î ╪▒╪º█î╪¼ **[150]** ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪»╪│╪¬█î╪º╪¿█î ╪¿┘ç 40-50% ┌⌐╪º┘ç╪┤ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ### ┘à╪▒╪º╪¼╪╣ [151-184]: ╪¬╪ú╪½█î╪▒ ┘ê ╪¼┘ç╪¬┘ç╪º█î ╪ó█î┘å╪»┘ç **[151]** ┘╛█î╪┤╪▒┘ü╪¬┘ç╪º█î ╪¿┘ç╪▒┘ç┘ê╪▒█î ╪¬┘ê╪│╪╣┘ç ╪º╪¼╪º╪▓┘ç ╪¬┘à╪▒┌⌐╪▓ ╪¿╪▒ ┘à┘å╪╖┘é ╪«╪º╪╡ ╪¡┘ê╪▓┘ç **[152]** ╪½╪¿╪º╪¬ ╪¿╪▒╪¬╪▒ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ ┘ê 70% ┘à╪│╪º╪ª┘ä ┌⌐█î┘ü█î ┌⌐┘à╪¬╪▒ **[153]** ┘à┘ä╪º╪¡╪╕╪º╪¬ ┘à┘å╪¡┘å█î █î╪º╪»┌»█î╪▒█î: 2-3 ┘ç┘ü╪¬┘ç ┘à┘ç╪º╪▒╪¬ ╪¿╪▒╪º█î ╪¬┘ê╪│╪╣┘ç╪»┘ç┘å╪»┌»╪º┘å ┘à╪»╪▒┘å C++ **[154]** ╪¬╪║█î█î╪▒╪º╪¬ ╪¿┘å█î╪º╪»█î ┘à╪╣┘à╪º╪▒█î ┌å╪º╪▒┌å┘ê╪¿ ╪¿╪º ╪▒┘ê█î┌⌐╪▒╪» ┘à╪¬╪º┌⌐┘ä╪º╪│-╪º┘ê┘ä **[155]** ╪▒╪º┘ç╪¡┘ä┘ç╪º█î ╪│╪▒█î╪º┘ä╪│╪º╪▓█î ╪¼┘ç╪º┘å█î ╪¿╪º ┌⌐╪º╪▒╪º█î█î ╪¿╪º┘ä╪º ┘à╪º┘å┘å╪» reflection\_serializer **[156]** ╪º╪¬╪╡╪º┘ä ╪«┘ê╪»┌⌐╪º╪▒ ╪«╪º╪╡█î╪¬ ╪¡╪░┘ü ┘╛█î┌å█î╪»┌»█î ╪»╪▒ ╪¬┘ê╪│╪╣┘ç ┌⌐╪º╪▒╪¿╪▒╪» **[157]** ╪¬┘é┘ê█î╪¬ ╪¬╪¡┘ä█î┘ä ╪º╪│╪¬╪º╪¬█î┌⌐ ╪¿╪º ╪¿█î┘å╪┤ ╪¿█î╪│╪º╪¿┘é┘ç ╪│╪º╪«╪¬╪º╪▒ ╪¿╪▒┘å╪º┘à┘ç **[158]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪│█î╪│╪¬┘à ╪│╪º╪«╪¬ ╪»╪│╪¬█î╪º╪¿█î ╪¿┘ç 20-30% ╪▓┘à╪º┘å ╪│╪º╪«╪¬ ╪│╪▒█î╪╣╪¬╪▒ **[159]** ╪¬┘ê┘ä█î╪» ╪«┘ê╪»┌⌐╪º╪▒ ┘à╪│╪¬┘å╪»╪º╪¬ ╪¼╪º┘à╪╣ API **[160]** ╪¬╪»╪▒█î╪│ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ╪╣┘à┘ê┘à█î ╪¿╪º 40% ╪»╪▒┌⌐ ╪│╪▒█î╪╣╪¬╪▒ ┘à┘ü┘ç┘ê┘à **[161]** ┘╛╪░█î╪▒╪┤ ╪╡┘å╪╣╪¬█î ╪»╪▒ ╪╡┘å╪º█î╪╣ ╪¿╪¡╪▒╪º┘å█î ┌⌐╪º╪▒╪º█î█î (╪¿╪º╪▓█î╪î ┘à╪º┘ä█î╪î ╪¬╪╣╪¿█î┘ç╪┤╪»┘ç) **[162]** ╪┤╪¬╪º╪¿ ┘à┘å╪¿╪╣ ╪¿╪º╪▓ ┘ê ┘å┘ê╪ó┘ê╪▒█î ╪»╪▒ ╪¼╪º┘à╪╣┘ç C++ **[163]** ╪º╪╡┘ä ┘à╪│╪ª┘ê┘ä█î╪¬ ┘ê╪º╪¡╪» ╪»╪▒ ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ **[164]** ╪¬╪▒┌⌐█î╪¿┘╛╪░█î╪▒█î ╪»╪▒ ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¿╪º ╪▒╪º╪¿╪╖┘ç╪º█î ┘ê╪º╪╢╪¡ **[165]** ╪ó┌»╪º┘ç█î ┌⌐╪º╪▒╪º█î█î ╪»╪▒ ╪╖╪▒╪º╪¡█î ┘à╪¬╪º┌⌐┘ä╪º╪│ ╪¬╪▒╪¼█î╪¡ ╪¬┘ê┘ä█î╪» ┌⌐╪» ╪│╪º╪»┘ç **[166]** ┘à╪»█î╪▒█î╪¬ ╪«╪╖╪º ╪¿╪º ┘╛█î╪º┘à┘ç╪º█î ╪«╪╖╪º█î ┘ê╪º╪╢╪¡ ┘é╪º╪¿┘ä ╪º╪¼╪▒╪º **[167]** ╪¬╪▒╪¼█î╪¡ ╪▓┘à╪º┘å ┌⌐╪º┘à┘╛╪º█î┘ä ╪¿╪▒ ╪▒┘ê█î┌⌐╪▒╪»┘ç╪º█î ╪▓┘à╪º┘å ╪º╪¼╪▒╪º ╪¿╪▒╪º█î ┌⌐╪º╪▒╪º█î█î ┘ê ╪º█î┘à┘å█î ┘å┘ê╪╣ **[168]** ╪º╪│╪¬╪▒╪º╪¬┌ÿ█î ┌⌐╪┤ ╪¿╪▒╪º█î ┘à╪¡╪º╪│╪¿╪º╪¬ ┌»╪▒╪º┘å ╪¿╪º╪▓╪¬╪º╪¿ **[169]** ╪º█î┘à┘å█î ┘å┘ê╪╣ ╪¿┘ç╪▒┘ç╪¿╪▒╪»╪º╪▒█î ╪º╪▓ ┘ê█î┌ÿ┌»█î┘ç╪º█î ╪¿╪º╪▓╪¬╪º╪¿ ╪¿╪▒╪º█î ╪¼┘ä┘ê┌»█î╪▒█î ╪º╪▓ ╪«╪╖╪º┘ç╪º█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î **[170]** ╪¬╪│╪¬ ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º ┘à╪¼┘à┘ê╪╣┘ç┘ç╪º█î ╪¬╪│╪¬ ╪«┘ê╪»┌⌐╪º╪▒ **[171]** ╪¬╪│╪¬ ╪▒┌»╪▒╪│█î┘ê┘å ┌⌐╪º╪▒╪º█î█î ╪¿╪º ┘å╪╕╪º╪▒╪¬ ┘à╪»╪º┘ê┘à **[172]** ╪º╪╣╪¬╪¿╪º╪▒╪│┘å╪¼█î ┌å┘å╪» ┌⌐╪º┘à┘╛╪º█î┘ä╪▒█î ╪¿╪▒╪º█î ┘é╪º╪¿┘ä█î╪¬ ╪¡┘à┘ä ╪»╪▒ ┘╛█î╪º╪»┘ç╪│╪º╪▓█î┘ç╪º **[173]** ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ╪¬╪ú█î█î╪» ╪▒╪│┘à█î ╪¿╪▒╪º█î ┌⌐╪» ╪¬┘ê┘ä█î╪» ╪┤╪»┘ç ╪¿╪º╪▓╪¬╪º╪¿ **[174]** ┌»╪│╪¬╪▒╪┤┘ç╪º█î ╪¬╪ª┘ê╪▒█î ┘å┘ê╪╣ ╪¿╪▒╪▒╪│█î ╪▒╪º╪¿╪╖┘ç ╪¿█î┘å ╪º┘å┘ê╪º╪╣ ┘ê ┘à╪¬╪º-╪º┘å┘ê╪º╪╣ **[175]** ╪¬╪¡┘ä█î┘ä ┘╛█î┌å█î╪»┌»█î ┘╛█î┌å█î╪»┌»█î ┘à╪¡╪º╪│╪¿╪º╪¬█î ╪╣┘à┘ä█î╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ **[176]** ╪¼╪º╪│╪º╪▓█î ╪▓╪¿╪º┘å┘ç╪º█î ╪«╪º╪╡ ╪¡┘ê╪▓┘ç ╪»╪▒ C++ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º **[177]** ╪¿┘ç█î┘å┘ç╪│╪º╪▓█î ╪«┘ê╪»┌⌐╪º╪▒ ╪¿╪º ╪º╪│╪¬┘ü╪º╪»┘ç ╪º╪▓ ╪¬┌⌐┘å█î┌⌐┘ç╪º█î █î╪º╪»┌»█î╪▒█î ┘à╪º╪┤█î┘å ╪¿╪º ┘à╪¬╪º╪»█î╪¬╪º█î ╪¿╪º╪▓╪¬╪º╪¿ **[178]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿█î┘å ╪▓╪¿╪º┘å┘ç ╪¬╪│┘ç█î┘ä █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿┘ç╪¬╪▒ C++ ╪¿╪º ╪│╪º█î╪▒ ╪▓╪¿╪º┘å┘ç╪º **[179]** ╪¬┌⌐╪º┘à┘ä ╪º╪▓ ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┘╛█î┌å█î╪»┘ç ╪¿┘ç ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪╣┘ä╪º┘å█î **[180]** ┘╛╪º╪▒╪º╪»╪º█î┘à ┌⌐╪» ╪¿┘ç ╪╣┘å┘ê╪º┘å ╪»╪º╪»┘ç ╪¿┘ç ╪┤█î┘ê┘ç ╪ó┌»╪º┘ç ╪º╪▓ ┌⌐╪º╪▒╪º█î█î **[181]** ╪»┘à┘ê┌⌐╪▒╪º╪¬█î┌⌐╪│╪º╪▓█î ╪¬┌⌐┘å█î┌⌐┘ç╪º█î ┘╛█î╪┤╪▒┘ü╪¬┘ç ┘é╪¿┘ä╪º┘ï ╪»╪▒ ╪»╪│╪¬╪▒╪│ ┘ü┘é╪╖ ┘à╪¬╪«╪╡╪╡╪º┘å **[182]** ┘å┌»╪º┘ç ╪¿█î╪│╪¬ ╪│╪º┘ä┘ç ╪¿┘ç ╪│┘à╪¬ ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘à╪¿╪¬┘å█î ╪¿╪▒ ╪¿╪º╪▓╪¬╪º╪¿ ╪¿┘ç ╪╣┘å┘ê╪º┘å ┘ç┘å╪¼╪º╪▒ **[183]** █î┌⌐┘╛╪º╪▒┌å┌»█î ╪¿╪º ╪¬┌⌐┘å┘ê┘ä┘ê┌ÿ█î┘ç╪º█î ┘å┘ê╪╕┘ç┘ê╪▒ ╪»╪▒ ╪│█î╪│╪¬┘à┘ç╪º█î ┘å╪º┘ç┘à┌»┘å╪î ╪¬┘ê╪▓█î╪╣ ╪┤╪»┘ç╪î ╪¬┘é┘ê█î╪¬ ╪┤╪»┘ç ╪¿╪º AI **[184]** ╪¬╪¡┘ê┘ä ╪ó┘à┘ê╪▓╪┤█î ╪¿╪º ┘à╪╣╪▒┘ü█î ╪▓┘ê╪»╪¬╪▒ ┘à┘ü╪º┘ç█î┘à ┘╛█î╪┤╪▒┘ü╪¬┘ç ╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ΓÇö ## ╪ó┘à╪º╪▒┘ç╪º█î ╪«┘ä╪º╪╡┘ç - **┌⌐┘ä ┘à╪▒╪º╪¼╪╣ █î╪º┘ü╪¬ ╪┤╪»┘ç:** 184 ╪º╪▓ 184 ╪░┌⌐╪▒ ╪┤╪»┘ç - **┘à╪▒╪º╪¼╪╣ ┌»┘à╪┤╪»┘ç:** ┘ç█î┌å - ┘ç┘à┘ç ┘à╪▒╪º╪¼╪╣ ╪¿╪º ┘à┘ê┘ü┘é█î╪¬ ┘à┌⌐╪º┘å█î╪º╪¿█î ┘ê ╪»╪│╪¬┘ç╪¿┘å╪»█î ╪┤╪»┘å╪» - **┘à╪¡╪»┘ê╪»┘ç ┘à╪▒╪º╪¼╪╣:** [1] ╪¬╪º [184] - **┘╛┘ê╪┤╪┤:** ╪┤╪º┘à┘ä ╪¬┘à╪º┘à 10 ╪¿╪«╪┤ ┘à┘é╪º┘ä┘ç ╪╣┘ä┘à█î - **╪»╪│╪¬┘ç┘ç╪º:** ┘à╪¿╪º┘å█î ┘ê ┘╛█î╪┤█î┘å┘ç (1-50)╪î ┌å╪º╪▒┌å┘ê╪¿ ┘ê ╪╖╪▒╪º╪¡█î (51-100)╪î ┌⌐╪º╪▒╪º█î█î ┘ê ┘à╪╖╪º┘ä╪╣╪º╪¬ ┘à┘ê╪▒╪»█î (101-150)╪î ╪¬╪ú╪½█î╪▒ ┘ê ╪ó█î┘å╪»┘ç (151-184) ## █î╪º╪»╪»╪º╪┤╪¬┘ç╪º ╪º█î┘å ┌⌐╪¬╪º╪¿╪┤┘å╪º╪│█î ╪º╪▓ ╪╖╪▒█î┘é ╪¬╪¡┘ä█î┘ä ╪│█î╪│╪¬┘à╪º╪¬█î┌⌐ ╪¬┘à╪º┘à ┘ü╪º█î┘ä┘ç╪º█î markdown ╪º┘å┌»┘ä█î╪│█î (01-10) ╪¡╪º┘ê█î ┘à┘é╪º┘ä┘ç ╪╣┘ä┘à█î ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪º╪│╪¬╪«╪▒╪º╪¼ ╪┤╪»┘ç ╪º╪│╪¬. ┘ç╪▒ ╪┤┘à╪º╪▒┘ç ┘à╪▒╪¼╪╣ ┘à╪╖╪º╪¿┘é ╪¿╪º ╪º╪▒╪¼╪º╪╣╪º╪¬ ╪»╪▒┘ê┘å╪«╪╖█î █î╪º┘ü╪¬ ╪┤╪»┘ç ╪»╪▒ ╪│╪▒╪º╪│╪▒ ╪¿╪«╪┤┘ç╪º█î ┘à┘é╪º┘ä┘ç ╪º╪│╪¬. ╪¬┘à╪º┘à 184 ┘à╪▒╪¼╪╣ ╪¿╪º ┘à┘ê┘ü┘é█î╪¬ ┘à┌⌐╪º┘å█î╪º╪¿█î ┘ê ╪»╪│╪¬┘ç╪¿┘å╪»█î ╪┤╪»┘ç╪º┘å╪». ┘à╪▒╪º╪¼╪╣ ╪╖█î┘ü ┌⌐╪º┘à┘ä█î ╪º╪▓ ╪¬╪¡┘é█î┘é╪º╪¬ ╪¿╪º╪▓╪¬╪º╪¿ ┘ê ┘à╪¬╪º┌⌐┘ä╪º╪│┘ç╪º█î C++23 ╪▒╪º ┘╛┘ê╪┤╪┤ ┘à█î╪»┘ç┘å╪»╪î ╪º╪▓ ╪¬┌⌐╪º┘à┘ä ╪¬╪º╪▒█î╪«█î ┘à╪¬╪º╪¿╪▒┘å╪º┘à┘ç┘å┘ê█î╪│█î ┘é╪º┘ä╪¿ ┌»╪▒┘ü╪¬┘ç ╪¬╪º ╪╖╪▒╪º╪¡█î ╪¿╪º╪▓╪¬╪º╪¿ ╪º╪│╪¬╪º╪¬█î┌⌐ ┘à╪»╪▒┘å╪î ╪¬╪¡┘ä█î┘ä ┌⌐╪º╪▒╪º█î█î╪î ┌⌐╪º╪▒╪¿╪▒╪»┘ç╪º█î ╪╣┘à┘ä█î╪î ┌å╪º┘ä╪┤┘ç╪º█î ┘╛█î╪º╪»┘ç╪│╪º╪▓█î ┘ê ╪¼┘ç╪¬┘ç╪º█î ╪¬╪¡┘é█î┘é ╪ó█î┘å╪»┘ç.