Applying Carefully Recurring Template Pattern in C++

There is various material effectively accessible for "How" and "What" on CRTP. So, I will address the "Where" part that CRTP Applicability.

by Vishal Chovatiya - Aug 18, 20 - Web Dev Zone - Tutorial

Carefully Recurring Template Pattern(CRTP) in C++ is defined a powerful technique and a static alternative to virtual functions. But at the same time, learning it may seem a bit weird at first. If you are like me, you probably spend a lot of time searching on the Net which simple little trick you need to provide a program on where CRTP is to date-to-day coding. And, if you are on Embedded Programmer, you may see CRTP usage in Boost libraries. Also, Boost.operators or Boost.iterator will also help for 90% of the cases for embedded processors are either not up to date with standard or dated.

There is various material effectively accessible for "How" and "What" on CRTP. So, I will address the "Where" part that CRTP Applicability.

CRTP and Static Polymorphism In C++

```
template<typename specific_product>
class product_interface
{
    void\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
};

class foo : public product_interface<foo>
{
    int\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
    int\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
};

class bar : public product_interface<bar>
{
    int\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
    int\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
    int\* const\* const\* const\*\*\*vtbl\*\*\*ptr\*\*\*;
};

template<typename specific_product>
void\* const\* const\*\*\*product\*\*\*product_factory(int\* const\*\*\*);

int\* const\*\*\*foo_vtbl\*\*\*;
int\* const\*\*\*bar_vtbl\*\*\*;

void\* const\*\*\*foo\*\*\*(int\* const\*\*\*);
void\* const\*\*\*bar\*\*\*(int\* const\*\*\*);

int main()
{
    int\* const\*\*\*const\*\*\*const\*\*\*
```