Template template arguments (C++ only)

A template argument for a template template parameter is the name of a class template.

When the compiler tries to find a template to match the template template argument, it only considers primary class templates. (A *primary template* is the template that is being specialized.) The compiler will not consider any partial specialization even if their parameter lists match that of the template template parameter. For example, the compiler will not allow the following code:

```
template<class T, int i> class A {
   int x;
};

template<class T> class A<T, 5> {
   short x;
};

template<template<class T> class U> class B1 { };

B1<A> c;
```

The compiler will not allow the declaration `B1<A> c`. Although the partial specialization of A seems to match the template template parameter U of B1, the compiler considers only the primary template of A, which has different template parameters than U.

The compiler considers the partial specializations based on a template template argument once you have instantiated a specialization based on the corresponding template template parameter. The following example demonstrates this:

```
#include <iostream>
#include <typeinfo>

using namespace std;

template<class T, class U> class A {
public:
   int x;
};

template<class U> class A<int, U> {
public:
   short x;
};

template<template<class T, class U> class V> class B {
   V<int, char> i;
   V<char, char> j;
};

B<A> c;

int main() {
   cout << typeid(c.i.x).name() << endl;
   cout << typeid(c.j.x).name() << endl;
}
```

The following is the output of the above example:

```
short
int
```

The declaration `V<int, char> i` uses the partial specialization while the declaration `V<char, char> j` uses the primary template.

Parent topic:

→ Template arguments (C++ only)

Related reference

→ Partial specialization (C++ only)

→ Template instantiation (C++ only)

Note: This document describes the syntax, semantics, and IBM z/OS® XL C/C++ implementation of the C and C++ programming languages. For a general-purpose C or C++ standard reference, see cppreference.com.