英特尔 oneAPI 是一个跨行业、开放、基于标准的统一的编程模型，它为跨 CPU、GPU、FPGA、专用加速器的开发者提供统一的体验。oneAPI 旨在简化跨多架构的开发过程，使开发人员能够更快地开发出高性能的应用程序。

要开始使用 oneAPI，您需要安装英特尔 oneAPI 工具包。您可以在英特尔官方网站上找到安装说明。安装完成后，您可以使用 setvars 脚本来配置开发环境。

接下来，您可以使用英特尔 oneAPI DPC++/C++ 编译器来编写程序。DPC++ 是一种基于 C++ 的语言，它支持数据并行性和异构编程。您可以使用 DPC++ 来编写能够在多种设备上运行的程序。

其中，DPC++ 是 Data Parallel C++（数据并行 C++）的首字母缩写，它是 Intel 为了将 SYCL 引入 LLVM 和 oneAPI 所开发的开源项目。SYCL 是一种高级别的编程模型，旨在提高各种加速设备上的编程效率。简单来说，它是一种跨平台的抽象层，用户不需要关心底层的加速器具体是什么，按照标准编写统一的代码就可以在各种平台上运行。

DPC++ 建立在 SYCL 和现代 C++ 语言之上，具体来说是建立在 C++17 标准之上的。它支持数据并行性和异构编程，可以让开发人员使用相同的编程环境来编写能够在多种设备上运行的程序。

下面是一个简单的 DPC++ 代码示例，它演示了如何在设备上执行一个简单的向量加法运算：

#include <CL/sycl.hpp>

using namespace sycl;

int main() {

constexpr int N = 1024;

int a[N], b[N], c[N];

for (int i = 0; i < N; i++) {

a[i] = i;

b[i] = i;

}

{

buffer<int, 1> a\_buf(a, range<1>(N));

buffer<int, 1> b\_buf(b, range<1>(N));

buffer<int, 1> c\_buf(c, range<1>(N));

queue q;

q.submit([&](handler &h) {

auto a = a\_buf.get\_access<access::mode::read>(h);

auto b = b\_buf.get\_access<access::mode::read>(h);

auto c = c\_buf.get\_access<access::mode::write>(h);

h.parallel\_for(range<1>(N), [=](id<1> i) {

c[i] = a[i] + b[i];

});

});

}

for (int i = 0; i < N; i++)

std::cout << c[i] << std::endl;

return 0;

}

复制

这段代码创建了三个缓冲区，分别用于存储输入向量 a 和 b，以及输出向量 c。然后，它创建了一个队列，并在队列上提交了一个 lambda 函数。这个 lambda 函数使用 parallel\_for 来执行向量加法运算。最后，程序输出结果向量 c 中的元素。