**C++与COStream共享变量**

为了方便修改COStream程序中的变量，以使C++用户能够更加灵活地使用流程序，COStream编译器允许COStream程序引用外部C++程序定义的全局变量。在COStream程序中对引用的变量附加extern标识符，并不进行初始化；在C++程序中，将被引用的变量定义在全局命名空间中，而后每次修改这些共享变量都将修改COStream程序中的变量。

全局变量的使用实例如下：

**COStream文件：**

extern int ExternValue;

composite Main(output stream<double x>Out, input stream<double x>In)

{

stream<double x> Out = FMRadioCore(In)()

{

work  
{

Out[0].x = In[0].x+ExternValue;

}

}

}

**C++文件:**

#include "RunCOStream.h"

using namespace COStream;

int ExternValue;

int main()

{

double \*p = new double[1379];

RunCOStream costream = new RunCOStream(p,1379);

ExternValue = 2;

costream.Run();

ExternValue = -1;

costream.Run();

}

如上面的代码所示,在COStream文件中引用了外部变量ExternValue，该变量在C++源文件中定义，用户可以在C++文件中修改ExternValue。

**例子**

composite MyCompositeCall(input stream<int x> In, output stream<int x> Out)

{

param

float f;

int x=1;

if(x>0){

stream<int x>G;

G=MyOperator(In)//2.内置operator定义

{

work

{

G[0].x = In[0].x + f;

}

window

{

In sliding(3,2);

Out tumbling(1);

}

};

}

}

composite Main(input stream<int x> In, stream<int x> In1, output stream<int x> Out)

{

param int x, int y;

int i,f;

int realparam;

stream<int x> G,G1,G3;//stream独立声明

if(x>0){

G= MyCompositeCall(In)(f);//其他composite调用

}

else

(G,G1)=MyOperatorx(In,In1)//内置operator定义

{

work

{

G[0].x = x+In[0].x+In1[0];

x += 1;

}

window

{

In sliding(3,2);

G tumbling(1);

}

};

//在非pipeline和splitjion结构中不允许出现循环控制语句内含有composite调用语句，只允许出现普通语句

f=100000.0;

for(i=1;i<3;i++)

f += 3;

//关于pipeline

G1=pipeline(G)

{

//普通C语句

int a,b;

x=(4+y)/2;

a =3;

b =5;

for(i=0;i<10;i++)//循环控制可以出现在pipeline结构中

if(x>0)//条件控制

add MyCompositeCall(realparam[i]);// composite-call必须是单输入单输出

else

add MyCompositeCall(realparam[i]);

add MyCompositeCall(realparam);

};

//splitjion

G3=splitjoin(G)

{

int t=12315;

split duplicate();

for(i=0;i<10;i++)//循环控制可以出现在pipeline结构中

if(t>0)//条件控制

add MyCompositeCall(realparam[i]);// composite-call必须是单输入单输出

else

add MyCompositeCall(realparam[i]);

add MyCompositeCall(realparam);

join roundrobin();

};

}