有了 C++的基础知识我们就能够很好地理解 G4 的程序结构了。

G4 是采用的 gcc 编译器，因此其程序结构是和 C++一样的。
首先包括有一个主程序 main，然后分别包含有子程序 src 和头文件 include
以及其他调用文件 others
G4 里面为了与 C++相区别，程序后缀都是.cc，头文件后缀都是.hh
其中头文件在.cc 里面写也没问题，但是那样看起来不方便，建议还是按照 c++的习惯一一对应比较好。
那么，关键问题是要进行一个模拟我们都需要写哪些 src 和 include 的？
下面我们首先看 G4 里面的几个基本类，这些基本类基本上是与 src 一一对应的。

G4RunManager——对应主程序
这个类在主程序中用以初始化模拟信息的，或者形象地说是用于连接子程序的，
而连接方式是通过 Set 函数来完成的
大家可以从$G4INSTALL/source/run/G4RunManager.hh 里面查看各种 Set 函数，如

```
  public: // with description
    inline void SetUserInitialization(G4VUserDetectorConstruction* userInit)
    { userDetector = userInit; }
    inline void SetUserInitialization(G4VUserPhysicsList* userInit)
    {
      physicsList = userInit;
      kernel->SetPhysics(userInit);
    }
```

可以说 G4RunManager 类是贯穿整个程序模拟过程的总线，因此一般说来只能有一个
而开始一次 Run 的信号则是通过 BeamOn 函数发出的，其格式是

```
    virtual void BeamOn(G4int n_event,const char* macroFile=0,G4int n_select=-1);
```

可以通过多次调用 BeamOn 来实现循环计算。

其余子类包括几何结构类、物理设定类、粒子发射器类（源描述类）、事件处理类、径迹处理类等。
这些类可以按照两种不同的分类方式分类，每种分类方式都可以分为两类。

子类按照重要性分为强制类和可选类。
其中几何结构类（DetectorConstruction）、物理设定类（PhysicsList）、
源描述类（PrimaryGenerator）都属于强制类，是必须有的，缺少任一个程序都无法运行。
而事件处理类（EventAction）、步数据处理类（SteppingAction）、径迹处理类（TrackingAction）、运行处理类（RunAction）
都属于可选类，用户可以通过设定这些类来获取感兴趣的信息，虽然没有这些类程序一样可以运行，但是如果一个都没有的话，
这样的模拟是没有意义的，除非是用来检验几何结构的完备性。

子类按照调用过程分为初始化类和用户干涉类
其中几何结构类（DetectorConstruction）、物理设定类（PhysicsList）属于初始化类，
这两个类在每次 Run 开始后就对模拟过程进行初始化，之后在粒子发射开始后（BeamOn

开始后）就不再继续调用了。
而其余的类则是在每发射一个粒子后都需要调用的，具体调用过程请复习第二讲中的模拟算法。

下面讲一个特殊的类。
G4UImanager——UI（user interface）管理器
这个类的英文全名是 user interface manager，顾名思义是用于人机交互的，用户可以通过这个类来向模拟过程发出“命令”，
这个类在一定程度上使用很简洁的命令语句代替了上述用户干涉类的功能。

需要说明的是，上面所说的几个类并没有包括所有可选类，根据每个用户的需求，用户可以有很多其他类。

下面以$G4INSTALL/example/novice/N01 的例子为例看一下这些类，
因为这个粒子仅为了说明 Geant4 的基本运用，所以这个例子里面只定义了 3 个强制类，没有使用可选类来输出感兴趣的信息。
但是通过 UI 管理器查看了粒子输运过程中的一些信息。

```
 // Construct the default run manager
  //首先是要创建一个运行管理器的实例，根据 C++的知识我们知道每个类都必须通过他的实例来进行操作
  G4RunManager* runManager = new G4RunManager;

  // set mandatory initialization classes
  //通过 Set 函数设置几何结构和物理过程，这两个类属于强制类，也属于初始化类
  G4VUserDetectorConstruction* detector = new ExN01DetectorConstruction;
  runManager->SetUserInitialization(detector);
  //
  G4VUserPhysicsList* physics = new ExN01PhysicsList;
  runManager->SetUserInitialization(physics);

  // set mandatory user action class
  //通过 Set 函数进行源描述，这个类属于强制类，也属于用户干涉类
  G4VUserPrimaryGeneratorAction* gen_action = new ExN01PrimaryGeneratorAction;
  runManager->SetUserAction(gen_action);

  // Initialize G4 kernel
  //Set 完成后，通过 Initialize()函数初始化 G4 内核
  runManager->Initialize();
  // Get the pointer to the UI manager and set verbosities
  //创建 UI 管理器，通过 UI 管理器设置运行、事件、径迹的可视化精细程度。
  G4UImanager* UI = G4UImanager::GetUIpointer();
  UI->ApplyCommand("/run/verbose 1");
  UI->ApplyCommand("/event/verbose 1");
```

```
UI->ApplyCommand("/tracking/verbose 1");
// Start a run
//通过 BeamOn()函数开始模拟，这里 macroFile 和 n_Select 都是使用了默认值
G4int numberOfEvent = 3;
runManager->BeamOn(numberOfEvent);
```