在上一篇文章中，在main函数中写了所有的内容，非常乱。C++是面向对象编程语言，那么就将这些内容封装起来。

这样的话，main函数就剩下4行。

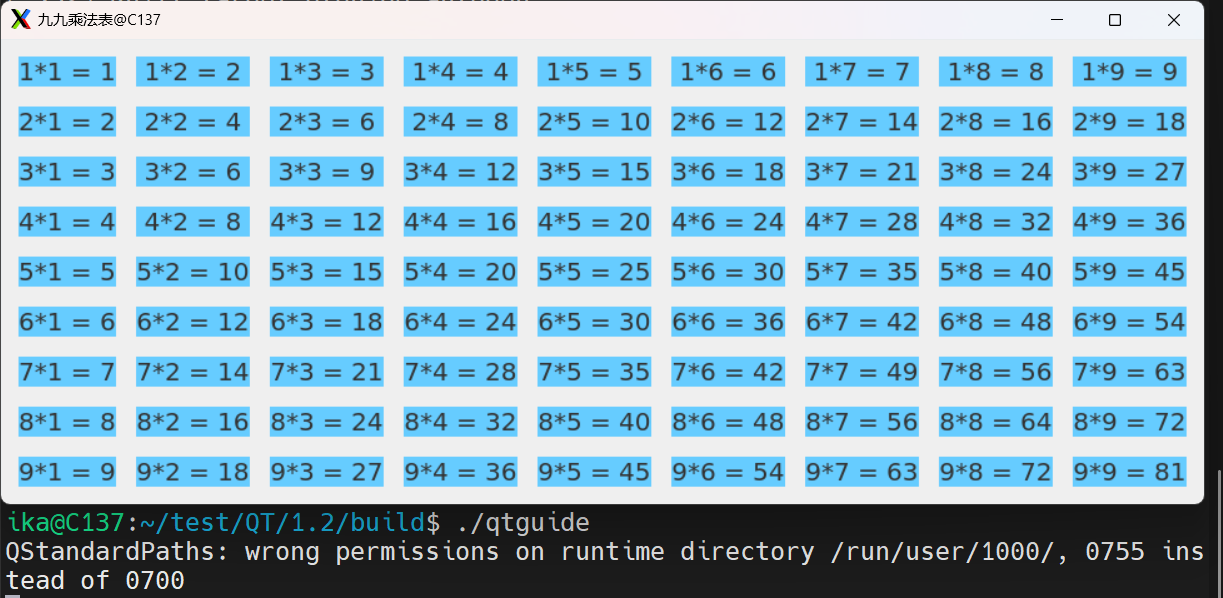
|  |
| --- |
| *int* main(*int* *argc*, *char* \*\**argv*){      QApplication app(*argc*, *argv*);      MainWindow window;      window.show();      return app.exec();  } |

首先是封装MainWindow

|  |
| --- |
| *class* MainWindow : *public* QMainWindow{  *public:*      explicit MainWindow(QWidget \**parent* = nullptr) : QMainWindow(*parent*){          setWindowTitle("九九乘法表");          setGeometry(500, 500, 200, 200);          //窗口的中心窗口，这个窗口是个QWidget，可以放置其他的QWidget，只有这一个窗口          QWidget \*centralWidget = new QWidget;          setCentralWidget(centralWidget);          //设置中心窗口的布局方式，这是个派生类，基类是QLayout，是个纯抽象类          //当然还有其他的布局方式，比如QHBoxLayout，QGridLayout等          QGridLayout \*layout = new QGridLayout();          centralWidget->setLayout(layout);          //QT提供的QString类，可以直接用+号连接字符串          for(*int* i = 1 ; i <= 9; i++)          for(*int* j = 1; j <= 9; j++){              Label \*label = new Label;              //label->setText("Hello" + QString::number(i));              label->setText(QString("%1 = %2").arg(QString::number(i) + "\*" + QString::number(j)).arg(i \* j));              layout->addWidget(label, i, j);          }      }  }; |

MainWindow继承了QT的QMainWindow，所有内容都写入构造函数中。构造函数接受一个参数。查看QMainWindow的源码，其构造函数是一样的。Explicit修饰以强调必须显示构造，构造函数的参数带有默认值nullptr（这样的话创建对象不传递值的话就会使用默认参数）。接下来在初始化列表中，初始基类和成员。初始化基类就是调用其构造函数。C++规定先构造基类，再构造派生类。如果将基类看作特殊的成员，那么派生类就构造完成了（现在派生类没有独有的成员）。

在这一版程序中，将QWidget设置为中心窗口（只能有一个）。然后，使用QT提供的layout在窗口中添加多个组件。布局的方式有水平、垂直和网格三种。这里使用网格窗口。



在MainWindow中，使用QString，比标准库的string多了很多功能。当然，QString也提供了和标准库string进行转换的成员函数。

然后是对QLabel的封装

|  |
| --- |
| *class* Label : *public* QLabel{  *public:*      explicit Label(QWidget \**parent* = nullptr) : QLabel(*parent*){          setAlignment(Qt::AlignCenter | Qt::AlignVCenter);            setStyleSheet(R"(          QLabel{          color: #333333;          font-size: 20px;          margin: 5px;          background-color: #66ccff          })");      }  }; |

比第一版的程序多了个margin，用来控制标签的间隙。

本篇完结。