```
/*
* Alabastra Project - C++ Editor writed with QT Library V.4
* Copyright (C) Igor Maculan - geocronos@gmail.com
    Marco Buoncristiano - marco.buoncristiano@gmail.com

* This program is free software; you can redistribute it and/or
* modify it under the terms of the GNU General Public License
* as published by the Free Software Foundation; either version 2
* of the License, or (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.

* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
*/
#include "AlWinMain.h"

#include <QStringList>
#include <QDebug>

using namespace std;

AlWinMain::AlWinMain()
{
 ui.setupUi(this);
 syntax=new AlSyntax(ui.txt);
 statusBar()->showMessage("Alabastra Started");
 new_file();
 QStringList word;
 word << "class" << "public" << "void" << "AlWinMain";
 c=new QCompleter(word,ui.txt);
 ui.txt->setCompleter(c);
 connect_action();
};

void AlWinMain::connect_action()
{
 connect
(ui.list_files,SIGNAL(itemClicked(QListWidgetItem*)),this,SLOT(view_file(QListWidgetIt
em*)));
 connect (ui.actionNuovo,SIGNAL(triggered()),this,SLOT(new_file()));
 connect (ui.actionSalva,SIGNAL(triggered()),this,SLOT(save()));
 connect (ui.actionSalva_con_nome,SIGNAL(triggered()),this,SLOT(save_as()));
 connect (ui.actionApri,SIGNAL(triggered()),this,SLOT(open()));
 connect (ui.actionChiudi,SIGNAL(triggered()),this,SLOT(close()));
 connect (ui.actionPdf,SIGNAL(triggered()),this,SLOT(filePdf()));
 connect (ui.actionAbout,SIGNAL(triggered()),this,SLOT(about()));
 //connect (ui.actionUndo,SIGNAL(triggered()),this,SLOT(undo()));
 connect (ui.txt,SIGNAL(cursorPositionChanged()),this,SLOT(showPosition()));

};

/**
```

```
*Crea un file nuovo
*/
void AlWinMain::new_file()
{
 AlDocument *dc=new AlDocument();
 ui.txt->setDocument(dc);
 ui.txt->setTabStopWidth(15);
 QListWidgetItem *lit;
 lit=new QListWidgetItem(dc->getFileName());
 connect (dc,SIGNAL( filenameChanged(AlDocument*) ),this,SLOT(
changeFileName(AlDocument*) ));
 QVariant t;
 t=qVariantFromValue((void*)dc);
 lit->setData(101,t);
 ui.list_files->addItem(lit);
 ui.list_files->setCurrentItem(lit);
 syntax->setDocument(ui.txt->document());
 ui.txt->setEnabled(true);
};

/**
*Restituisce il documento relativo al'item della lista passatogli come argomento
*/
AlDocument* AlWinMain::getDocumentFromList(QListWidgetItem *lit)
{
 QVariant d=lit->data(101);
 AlDocument *dc;
 dc=(AlDocument*)d.value<void*>();
 return dc;
};

/*
* Restituisce il documento attuale
*/
AlDocument* AlWinMain::getActualDocument(){
 return getDocumentFromList(ui.list_files->currentItem());
};

/**
*Visualizza il contenuto del file selezionato
*/
void AlWinMain::view_file(QListWidgetItem *lit){
 AlDocument *dc=getDocumentFromList(lit);
 ui.txt->setDocument(dc);
 syntax->setDocument(dc);
 ui.txt->setTextCursor(dc->getCursor());
};



/**
*Salva l'attuale file
*/

void AlWinMain::save(){
 save(getActualDocument());
};
```

```
void AlWinMain::save(AlDocument *dc)
{
 switch ( dc->save() )
 {
  case -1://ERRORE DI sALVATAGGIO
    QMessageBox::critical(this,tr("Salvataggio File"),tr("Salvataggio non riuscito"));
    break;
  case 0: break; //SAVE OK
  case 1: //MAI Salvato
    save_as();
    break;
  case 2:break; //NON MODIFICATO
 }
};

/**
*Salva con nome l'attuale documento
*/
void AlWinMain::save_as()
{
 QString fname=QFileDialog::getSaveFileName(this,tr("Salva con nome"),"","Source
(*.cpp *.c);;Header (*.h)");
 if (!fname.isEmpty())
 {
  //Salvo il file
  if ( ! getActualDocument()->save(fname) )
    QMessageBox::critical(this,tr("Salvataggio File"),tr("Salvataggio non riuscito"));
 }
};

/**
 Evento scatenato dal cambio del nome file (SALVA)
*/
void AlWinMain::changeFileName(AlDocument *dc)
{
 ui.list_files->currentItem()->setText(dc->getFileName());
};

/**
 Apertura File
*/
void AlWinMain::open()
{
 QString fname=QFileDialog::getOpenFileName(this,tr("Apri"),"","Source (*.cpp
*.c);;Header (*.h)");
 if (!fname.isEmpty()){
  //Apro il file
  AlDocument *dc=getActualDocument();
  if ( !dc->isSaved() && !dc->isModified())
    dc->open(fname);
  else{
    //Creo un nuovo file
    new_file();
    dc=getActualDocument();
    dc->open(fname);
  }
```

```
 }
};

/**
 Chiusura File
*/
void AlWinMain::close(QListWidgetItem *lit){
 AlDocument *dc=getDocumentFromList(lit);
 int row=ui.list_files->row(lit);
 if (dc->isModified() || (!dc->isSaved() && !dc->isEmpty()))
 {
  int r=QMessageBox::question(this,tr("Chiusura File"),tr("Si vuole salvare le
modifiche prima di chiudere il file?"),tr("Si"),tr("No"),tr("ANNULLA"));
  switch(r){
   case 0:{
      save();
      if (dc->isSaved() && !dc->isModified())
       ui.list_files->takeItem(row);
     }break;
   case 1:{
      ui.list_files->takeItem(row);
     }break;
   default:
    return;
  }
 }
 else
  ui.list_files->takeItem(row);

 //se chiudo l'unico file mi posiziono sul 1o
 if (ui.list_files->count()==0){
  ui.txt->clear();
  ui.txt->setEnabled(false);
 }else{
  //mi posiziono su un'altro file della lista
  int rn;
  if (row-1>0)
   rn=row-1;
  else
   rn=row;
  ui.list_files->setCurrentRow(rn);
  view_file(ui.list_files->currentItem());
 }
};

/*
*chiude il file attuale
*/
void AlWinMain::close(){
 close(ui.list_files->currentItem());
};

/*
*Visualizza la riga e la colonna in fondo all'editor
*/
void AlWinMain::showPosition(){
 ui.lbl_bottom->setText(tr("Riga: %1 Colonna: %2")
```

```
 .arg(ui.txt->textCursor().blockNumber())
 .arg(ui.txt->textCursor().columnNumber()) );
};

/*
* Alla chiusura del programma
*/
void AlWinMain::closeEvent(QCloseEvent *e){
 AlDocument *dc;
 QListWidgetItem *lit;
 bool annulla=false;
 while (ui.list_files->count()>0 && !annulla) {
  lit=ui.list_files->item(0);
  dc=getDocumentFromList(lit);
  if (dc->isSaved())
   close(lit);
  else{
   int r=QMessageBox::question(this,tr("Chiusura File
%1").arg(dc->getFileName()),tr("Si vuole salvare le modifiche prima di chiudere il
file?"),tr("Si"),tr("No"),tr("ANNULLA"));
   switch (r){
    case 0:{
     save(dc);
     if (dc->isSaved() && !dc->isModified())
      ui.list_files->takeItem(0);
     else
      annulla=true;
     break;
    }
    case 1:{
     ui.list_files->takeItem(0);
     break;
    }
    case 2:{
     annulla=true;
     break;
    }
   }
  }
 }
 if (!annulla)
  e->accept();
 else
  e->ignore();
};

//esporta il file corrente in formato pdf
void AlWinMain::filePdf()
{
#ifndef QT_NO_PRINTER
    QString fname = QFileDialog::getSaveFileName(this, "Esporta in PDF", QString(),
"*.pdf");
    if (!fname.isEmpty()) {//se vuoto
        if (QFileInfo(fname).suffix().isEmpty())
            fname.append(".pdf");
        QPrinter printer(QPrinter::HighResolution);
        printer.setOutputFormat(QPrinter::PdfFormat);
```

```
        printer.setOutputFileName(fname);
        ui.txt->document()->print(&printer);
    }
#endif
};

void AlWinMain::about()
{

   QMessageBox::about(this, tr("Alabastra "),
           tr(" <b>Alabastra</b>  un editor scritto interamente in Qt"));

};
```