main.cpp

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char **argv)
{
	QApplication app(argc, argv); // Create a QApplication object
	MaFenetre maFenetre; // Create a window object
	maFenetre.show(); // Show the window
	int ret = app.exec(); // Execute the main event loop
	return ret;
}
```

mainwindow.cpp

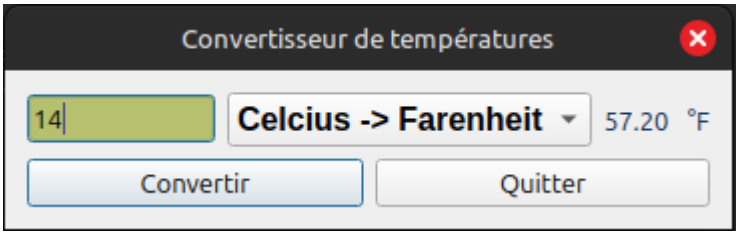
```
#include "mainwindow.h"

MaFenetre::MaFenetre(QWidget *parent) : QDialog(parent)
{
	// 1. Instancier les widgets
	valeur = new QLineEdit(this);
	resultat = new QLabel(this);
	unite = new QLabel(this);
	choixConversion = new QComboBox(this);
	bConvertir = new QPushButton(QString::fromUtf8("Convertir"),
this);
	bQuitter = new QPushButton(QString::fromUtf8("Quitter"), this);

	// 2. Personnaliser les widgets
	valeur->setStyleSheet("color: #000000; background-color:
#b6c06e;");
	valeur->clear();
	QFont font("Liberation Sans", 12, QFont::Bold);
	choixConversion->setFont(font);
	choixConversion->addItem(QString::fromUtf8("Celcius ->
Farenheit"));
	choixConversion->addItem(QString::fromUtf8("Farenheit ->
Celcius"));
	resultat->setStyleSheet("color: #0a214c;");
	unite->setStyleSheet("color: #0a214c;");

	// 3. Instancier les layouts
	QHBoxLayout *hLayout1 = new QHBoxLayout;
	QHBoxLayout *hLayout2 = new QHBoxLayout;
```

```
      QVBoxLayout *mainLayout = new QVBoxLayout;

      // 4. Positionner les widgets dans les layouts
      hLayout1->addWidget(valeur);
      hLayout1->addWidget(choixConversion);
      hLayout1->addWidget(resultat);
      hLayout1->addWidget(unite);
      hLayout2->addWidget(bConvertir);
      hLayout2->addWidget(bQuitter);
      mainLayout->addLayout(hLayout1);
      mainLayout->addLayout(hLayout2);
      setLayout(mainLayout);

      // 5. Connecter les signaux et slots (using new syntax)
      connect(bConvertir, SIGNAL(clicked()), this, SLOT(convertir()));
      connect(this, SIGNAL(actualiser()), this, SLOT(convertir()));
      connect(choixConversion, SIGNAL(currentIndexChanged(int)),
this,SLOT(permuter()));
      connect(bQuitter, SIGNAL(clicked()), qApp, SLOT(quit()));
      // bonus : conversion automatique
      connect(valeur, SIGNAL(textChanged(const QString &)), this,
SLOT(convertir()));

      // 6. Personnaliser la fenêtre
      setWindowTitle(QString::fromUtf8("Convertisseur de
températures"));
      adjustSize();

      // Start conversion
      emit actualiser();
}

// 7. Définir les slots
void MaFenetre::convertir()
{
      QString temperature = valeur->text();
      if (temperature.isEmpty())
      {
      resultat->setText(QString::fromUtf8("--.--"));
      afficherUnite();
      return;
      }

      bool ok;
      double tempValue = temperature.toDouble(&ok);
      if (!ok) {
```

```
      resultat->setText(QString::fromUtf8("Erreur"));
      afficherUnite();
      return;
      }

      switch (choixConversion->currentIndex())
      {
      case CELCIUS_FARENHEIT:
      resultat->setText(QString::fromUtf8("%1").arg(9 * tempValue / 5 +
32, 0, 'f', 2));
      break;
      case FARENHEIT_CELCIUS:
      double celsius = 5 * (tempValue - 32) / 9;
      resultat->setText(QString::number(celsius, 'f', 2));
      break;
      }
}

void MaFenetre::permuter()
{
      if (resultat->text() != "--.--")
      {
      valeur->setText(resultat->text());
      emit actualiser();
      }
      afficherUnite();
}

// 8. Définir les méthodes
void MaFenetre::afficherUnite()
{
      switch (choixConversion->currentIndex())
      {
      case CELCIUS_FARENHEIT:
      unite->setText(QString::fromUtf8(" °F"));
      break;
      case FARENHEIT_CELCIUS:
      unite->setText(QString::fromUtf8(" °C"));
      break;
      }
}
```

mainwindow.h

```
#include <QApplication>
#if QT_VERSION >= 0x050000
#include <QtWidgets> /* tous les widgets de Qt5 */
#else
#include <QtGui> /* tous les widgets de Qt4 */
#endif
#define CELCIUS_FARENHEIT 0
#define FARENHEIT_CELCIUS 1
class MaFenetre : public QDialog
{
      Q_OBJECT
      // Membre(s) public(s)
public:
      MaFenetre( QWidget *parent = 0 );
      // Membre(s) privé(s)
private:
      // Les widgets
      QLineEdit *valeur;
      QLabel *resultat;
      QLabel *unite;
      QComboBox *choixConversion;
      QPushButton *bConvertir;
      QPushButton *bQuitter;
      QDoubleValidator *doubleValidator;
      void afficherUnite();
      // Mécanisme(s) Qt
signals:
      void actualiser();
private slots:
      void convertir();
      void permuter();
} ;
```



mainwindow.cpp

```
#include "mainwindow.h"

MaFenetre::MaFenetre(QWidget *parent) : QDialog(parent)
{
      // 1. Instancier les widgets
      valeur = new QLineEdit(this);
      resultat = new QLabel(this);
      unite = new QLabel(this);
      choixConversion = new QComboBox(this);
      bConvertir = new QPushButton(QString::fromUtf8("Convertir"),
this);
      bQuitter = new QPushButton(QString::fromUtf8("Quitter"), this);

      // 2. Personnaliser les widgets
      valeur->setStyleSheet("color: #000000; background-color:
#b6c06e;");
      valeur->clear();
      QFont font("Liberation Sans", 12, QFont::Bold);
      choixConversion->setFont(font);
      choixConversion->addItem(QString::fromUtf8("Décimal -> Binaire"));
      choixConversion->addItem(QString::fromUtf8("Binaire -> Décimal"));
      resultat->setStyleSheet("color: #0a214c;");
      unite->setStyleSheet("color: #0a214c;");

      // 3. Instancier les layouts
      QHBoxLayout *hLayout1 = new QHBoxLayout;
      QHBoxLayout *hLayout2 = new QHBoxLayout;
      QVBoxLayout *mainLayout = new QVBoxLayout;

      // 4. Positionner les widgets dans les layouts
      hLayout1->addWidget(valeur);
      hLayout1->addWidget(choixConversion);
      hLayout1->addWidget(resultat);
      hLayout1->addWidget(unite);
      hLayout2->addWidget(bConvertir);
      hLayout2->addWidget(bQuitter);
      mainLayout->addLayout(hLayout1);
      mainLayout->addLayout(hLayout2);
      setLayout(mainLayout);

      // 5. Connecter les signaux et slots (using new syntax)
      connect(bConvertir, SIGNAL(clicked()), this, SLOT(convertir()));
      connect(this, SIGNAL(actualiser()), this, SLOT(convertir()));
      connect(choixConversion, SIGNAL(currentIndexChanged(int)),
```

```
this,SLOT(permuter()));
	connect(bQuitter, SIGNAL(clicked()), qApp, SLOT(quit()));
	// bonus : conversion automatique
	connect(valeur, SIGNAL(textChanged(const QString &)), this,
SLOT(convertir()));

	// 6. Personnaliser la fenêtre
	setWindowTitle(QString::fromUtf8("Convertisseur décimal
binaire"));
	adjustSize();

	// Start conversion
	emit actualiser();
}

// 7. Définir les slots
void MaFenetre::convertir()
{
	QString input = valeur->text();
	if (input.isEmpty())
	{
	resultat->setText(QString::fromUtf8("--.--"));
	afficherUnite();
	return;
	}

	bool ok;
	double decimalValue = input.toDouble(&ok);
	if (!ok)
	{

	bool isBinary = true;
	for (const QChar &ch : input) {
		if (ch != '0' && ch != '1') {
			isBinary = false;
			break;
		}
	}

	if (isBinary) {

		decimalValue = input.toLongLong(&ok, 2);
		if (ok) {
			resultat->setText(QString::number(decimalValue));
			afficherUnite();
			return;
```

```
            }
      }

      resultat->setText(QString::fromUtf8("Erreur"));
      afficherUnite();
      return;
      }

      switch (choixConversion->currentIndex())
      {
      case DECIMAL_TO_BINARY:
      resultat->setText(QString::number(static_cast<long
long>(decimalValue), 2));
      break;
      case BINARY_TO_DECIMAL:
      decimalValue = input.toLongLong(&ok, 2);
      if (ok) {
            resultat->setText(QString::number(decimalValue));
      } else {
            resultat->setText(QString::fromUtf8("Erreur"));
      }
      break;
      default:
      resultat->setText(QString::fromUtf8("Erreur"));
      break;
      }
}


void MaFenetre::permuter()
{
      if (resultat->text() != "--.--")
      {
      valeur->setText(resultat->text());
      emit actualiser();
      }
      afficherUnite();
}

// 8. Définir les méthodes
void MaFenetre::afficherUnite()
{
      switch (choixConversion->currentIndex())
      {
      case DECIMAL_TO_BINARY:
      unite->setText(QString::fromUtf8(" B²"));
```

```
        break;
        case BINARY_TO_DECIMAL:
        unite->setText(QString::fromUtf8(" D10"));
        break;
        }
}
```

main.cpp

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char **argv)
{
        QApplication app(argc, argv); // Create a QApplication object
        MaFenetre maFenetre; // Create a window object
        maFenetre.show(); // Show the window
        int ret = app.exec(); // Execute the main event loop
        return ret;
}
```

mainwindow.h

```
#include <QApplication>
#if QT_VERSION >= 0x050000
#include <QtWidgets> /* tous les widgets de Qt5 */
#else
#include <QtGui> /* tous les widgets de Qt4 */
#endif
#define DECIMAL_TO_BINARY 0
#define BINARY_TO_DECIMAL 1
class MaFenetre : public QDialog
{
        Q_OBJECT
        // Membre(s) public(s)
public:
        MaFenetre( QWidget *parent = 0 );
        // Membre(s) privé(s)
private:
        // Les widgets
        QLineEdit *valeur;
        QLabel *resultat;
        QLabel *unite;
        QComboBox *choixConversion;
        QPushButton *bConvertir;
        QPushButton *bQuitter;
        QDoubleValidator *doubleValidator;
```

```
      void afficherUnite();
      // Mécanisme(s) Qt
signals:
      void actualiser();
private slots:
      void convertir();
      void permuter();
} ;
```