Manual del KDevelop

Part d'aquesta documentació ha estat convertida des de la pàgina KDevelop4/Manual de KDE UserBase.
Traductor: Antoni Bella

Índex

1 Què és el KDevelop? 6

2 Sessions i projectes: Els fonaments del KDevelop 8

2.1 Terminologia . . . 8

2.2 Configurar una sessió i importar un projecte existent . . . 9

2.2.1 Opció 1: Importar un projecte des d'un servidor amb el sistema pel control de versions . . . 9

2.2.2 Opció 2: Importar un projecte que ja es troba al disc dur . . . 10

2.3 Configurar una aplicació com un segon projecte . . . 10

2.4 Crear projectes a partir de zero . . . 10

3 Treballar amb el codi font 12

3.1 Eines i vistes . . . 12

3.2 Explorar el codi font . . . 14

3.2.1 Informació local . . . 14

3.2.2 Informació sobre l'àmbit dels fitxers . . . 16

3.2.3 Informació sobre l'àmbit del projecte i la sessió . . . 17

3.2.4 Explicació del ressaltat amb els colors de l'Arc de Sant Martí . . . 19

3.3 Navegar pel codi font . . . 19

3.3.1 Navegació local . . . 19

3.3.2 Navegació en l'àmbit de fitxers i mode d'esquema . . . 20

3.3.3 Navegar sobre l'àmbit del projecte i la sessió: Navegació semàntica . . . 21

3.4 Escriure codi font . . . 25

3.4.1 Compleció automàtica . . . 25

3.4.2 Afegir classes noves i implementar funcions de membre . . . 27

3.4.3 Documentar les declaracions . . . 31

3.4.4 Canviar el nom de les variables, funcions i classes . . . 34

3.4.5 Retalls de codi . . . 35

3.5 Modes i conjunts de treball . . . 37

3.6 Algunes dreceres de teclat útils . . . 39

4 Generar el codi amb plantilles **41**
4.1 Crear una classe nova . . . 41
4.2 Crear una prova unitària nova . . . 43
4.3 Altres fitxers . . . 43
4.4 Gestionar les plantilles . . . 44

5 Construir (compilar) projectes amb els «Makefile» personalitzats **46**
5.1 Construir cadascun dels objectius «Makefile» . . . 46
5.2 Seleccionar un conjunt d'objectius «Makefile» per a repetir la construcció . . . 47
5.3 Què fer amb els missatges d'error . . . 48

6 Executar programes al KDevelop **49**
6.1 Configurar els llançaments al KDevelop . . . 49
6.2 Algunes dreceres de teclat útils . . . 50

7 Depurar programes al KDevelop **52**
7.1 Executar un programa en el depurador . . . 52
7.2 Adjuntar el depurador a un procés en execució . . . 53
7.3 Algunes dreceres de teclat útils . . . 54

8 Treballar amb sistemes de control de versions **55**

9 Personalitzar el KDevelop **57**
9.1 Personalitzar l'editor . . . 57
9.2 Personalitzar el sagnat del codi . . . 57
9.3 Personalitzar les dreceres del teclat . . . 59
9.4 Personalitzar la compleció automàtica del codi . . . 59

10 Crèdits i llicència **61**

Resum

El KDevelop és un sistema de desenvolupament integrat per a emprar-lo per a una àmplia varietat de tasques de programació.

Capítol 1

Què és el KDevelop?

El KDevelop és un entorn de desenvolupament integrat modern (IDE) per a C++ (i altres llenguatges) i és una de les moltes aplicacions del KDE. Com a tal, s'executa en Linux® (fins i tot si l'executeu en un dels altres escriptoris, com ara el GNOME), però també està disponible per a la majoria de les altres variants d'UNIX® i també per a Windows.

El KDevelop ofereix totes les comoditats dels IDE moderns. Per a grans projectes i aplicacions, la característica més important és que el KDevelop *entén* C++: analitza la totalitat de la base de la font i recorda quines classes tenen funcions de membre, on es defineixen les variables, de quins tipus són, i moltes altres coses sobre el vostre codi. Per exemple, diguem que un dels fitxers de capçalera del vostre projecte declara una classe

```
class Car {
  // ...
  public:
    std::string get_color () const;
};
```

i més endavant en el vostre programa teniu

```
Car my_ride;
// ...do something with this variable...
std::string color = my_ride.ge
```

caldrà recordar que `my_ride` en l'última línia és una variable del tipus `Car` i us oferirà completar `ge` com a `get_color()` ja que aquesta és l'única funció de membre de la classe `Car` que comença així. En comptes de continuar escrivint simplement premeu **Retorn** per a obtenir la paraula completa; això estalvia temps, evita errors ortogràfics, i no requereix recordar els noms exactes dels centenars o milers de funcions i classes que formen els grans projectes.

Com a segon exemple, suposeu que teniu un codi com aquest:

```
double foo ()
{
  double var = my_func();
 return var * var;
}
double bar ()
{
  double var = my_func();
  return var * var * var;
}
```

Si passeu el ratolí per sobre del símbol `var` a la funció `bar`, obtindreu l'opció de veure tots els usos d'aquest símbol. En fer clic tan sols es mostraran els usos d'aquesta variable en funció de `bar`, perquè el KDevelop entén que la variable `var` a la funció `foo` no té res a veure. De manera similar, fent clic dret sobre el nom de la variable permet canviar el nom d'aquesta variable -el fet de fer-ho, només tocarà la variable a `bar`, però no una amb el mateix nom a `foo`-.

Però el KDevelop no és només un editor intel·ligent de codi. Hi ha altres coses que el KDevelop fa bé. Òbviament, ressalta el codi font en diferents colors, disposa d'un sagnat adaptable, té una interfície integrada pel depurador gdb de GNU, podeu visualitzar la documentació d'una funció passant el ratolí per sobre d'un ús d'aquesta funció, podeu tractar amb diferents tipus d'entorns de compilació i compiladors (p. ex., amb projectes basats en **make** i **cmake**), i moltes altres coses interessants que es discuteixen en aquest manual.

Capítol 2

Sessions i projectes: Els fonaments del KDevelop

En aquesta secció repassarem alguna de la terminologia de la forma en què el KDevelop veu el món i com treballa de forma estructurada. En concret, presentarem el concepte de *sessions* i *projectes*, i explicarem com podeu configurar els projectes en els quals voleu treballar des del KDevelop.

2.1 Terminologia

El KDevelop té el concepte de *sessions* i *projectes*. Una sessió conté tots els projectes que tenen alguna cosa a veure entre si. Per als exemples que segueixen, s'assumeix que sou el desenvolupador tant d'una biblioteca com de l'aplicació que la utilitza. Podeu pensar primer en les biblioteques del KDE i després en el KDevelop. Un altre exemple: Diguem que sou un hacker del nucli Linux®, però també esteu treballant en un controlador de dispositiu per a Linux® que encara no ha estat fusionat a l'arbre del nucli.

De manera que prenent l'últim com a exemple, tindríeu una sessió al KDevelop que té dos projectes: el nucli Linux® i el controlador de dispositiu. Heu d'agrupar-los en una sola sessió (en lloc de tenir dues sessions amb un sol projecte cadascuna), ja que serà útil per a poder veure les funcions del nucli i l'estructuració de les dades al KDevelop sempre que escriviu codi font pel controlador -p. ex., per a obtenir la funció del nucli i els noms de les variables expandides automàticament, o per a poder veure la documentació d'aquesta funció del nucli mentre pirategeu el controlador del dispositiu-.

Ara imagineu-vos que també sou un desenvolupador del KDE. Llavors tindríeu una segona sessió que conté el KDE com un projecte. Es podria, en principi, tenir una sola sessió per a tot això, però no hi ha cap motiu real: en el vostre treball al KDE, no cal accedir a les funcions del nucli o als controladors de dispositiu -i no voleu que els noms de les classes del KDE s'expandeixin automàticament mentre treballeu en el nucli de Linux®-. Finalment, la construcció d'algunes de les biblioteques del KDE és independent de tornar a compilar el nucli de Linux® (el qual sempre que compileu el controlador del dispositiu també seria bo tornar a compilar el nucli de Linux® si també s'han canviat alguns dels fitxers de capçalera del nucli).

Finalment, un altre ús per a les sessions és si treballeu tant en la versió de desenvolupament d'un projecte, així com en una branca d'aquest: en aquest cas, no voleu que el KDevelop confongui les classes que pertanyen a la línia principal i les de la branca, de manera que tindreu dues sessions, amb el mateix joc de projectes, però des de directoris diferents (els quals es corresponen amb diferents branques de desenvolupament).

2.2 Configurar una sessió i importar un projecte existent

Seguirem amb l'exemple del nucli Linux® i el controlador del dispositiu -possiblement voldreu substituir el vostre propi conjunt de biblioteques o projectes per aquests dos exemples. Per s crear una sessió nova que contingui aquests dos projectes aneu al menú **Sessió** → **Inicia una sessió nova** a la part superior esquerra (o, si aquesta és la primera vegada que utilitzeu el KDevelop: simplement empreu la sessió per omissió que s'obté en el primer ús, la qual és buida).

A continuació, volem poblar aquesta sessió amb els projectes que de moment assumim que existeixen en algun lloc (el cas on es comencen projectes des de zero es discuteix en una altra part d'aquest manual). Per a això, bàsicament hi ha dos mètodes, depenent de si el projecte es troba en algun lloc del disc dur o si necessiteu descarregar-lo des d'un servidor.

2.2.1 Opció 1: Importar un projecte des d'un servidor amb el sistema pel control de versions

Primer assumirem que el projecte que volem configurar -el nucli de Linux®- resideix en algun sistema pel control de versions a un servidor, però encara no s'ha comprovat el vostre disc dur local. En aquest cas, aneu al menú **Projecte** per a crear el nucli Linux® com un projecte dins de la sessió actual i després feu els següents passos:

- Aneu a **Projecte** → **Recupera un projecte** per a importar un projecte.
- A continuació, teniu múltiples opcions per a iniciar un projecte nou en la sessió actual, depenent d'on provinguin els fitxers font: Simplement, podeu apuntar el KDevelop a un directori existent (vegeu l'opció 2 a continuació), o podeu demanar al KDevelop que obtingui les fonts des d'un repositori.
- Assumint que no teniu una versió descarregada:
  - En el diàleg, sota **Seleccioneu l'origen**, escolliu emprar **Des del sistema de fitxers**, **Subversion**, **Git**, **GitHub** o **KDE**.
  - Escolliu un directori de treball com a destinació en el qual es descarregaran les fonts.
  - Escolliu un URL per a la ubicació del repositori des d'on es podran obtenir els fitxers d'origen.
  - Premeu **Obtén**. Això pot trigar molta estona, depenent de la velocitat de la vostra connexió i la mida del projecte. Malauradament, al KDevelop 4.2.x la barra de progrés en realitat no mostra res, però podeu seguir el progrés consultant periòdicament la sortida de la línia d'ordres per a veure la quantitat de dades que ja s'han descarregat.

> NOTA
> El problema amb la barra de progrés ha estat informat com a error 256832 del KDevelop.

> NOTA
> Durant aquest procés, també obtinc el missatge *Heu d'especificar una ubicació vàlida del projecte*, el qual es pot ignorar sense problemes.

- Se us demanarà que seleccioneu un fitxer de projecte del KDevelop en aquest directori. Ja que probablement encara no en teniu un, simplement premeu **Següent**.
- Premeu **Següent** una altra vegada.

- El KDevelop us demanarà que seleccioneu un gestor del projecte. Si aquest projecte empra els fitxers make estàndards a UNIX®, escolliu el gestor del projecte «makefile» personalitzat.
- A continuació, el KDevelop començarà a analitzar tot el projecte. Una vegada més, trigarà força temps per a anar a través de tots els fitxers i l'índex de les classes, etc. A la part inferior dreta de la finestra principal, hi ha una barra de progrés que mostra el temps que pot trigar aquest procés. (Si teniu diversos nuclis de processador, podeu accelerar aquest procés anant a l'element de menú **Arranjament** → **Configura el KDevelop...**, després seleccionant **Analitzador en segon pla** a l'esquerra, i incrementant el nombre de fils per a analitzar en segon pla a la dreta).

2.2.2 Opció 2: Importar un projecte que ja es troba al disc dur

De manera alternativa, si el projecte sobre el qual voleu treballar ja existeix al vostre disc dur (p. ex., pel fet que l'heu descarregat com un fitxer TAR des d'un servidor FTP, atès que ja heu obtingut una versió nova del projecte des d'un sistema pel control de versions, o perquè és el vostre propi projecte que *només* existeix al vostre disc dur), tot seguit utilitzeu **Projectes** → **Obre / importa un projecte** i en el diàleg trieu el directori on resideix el vostre projecte.

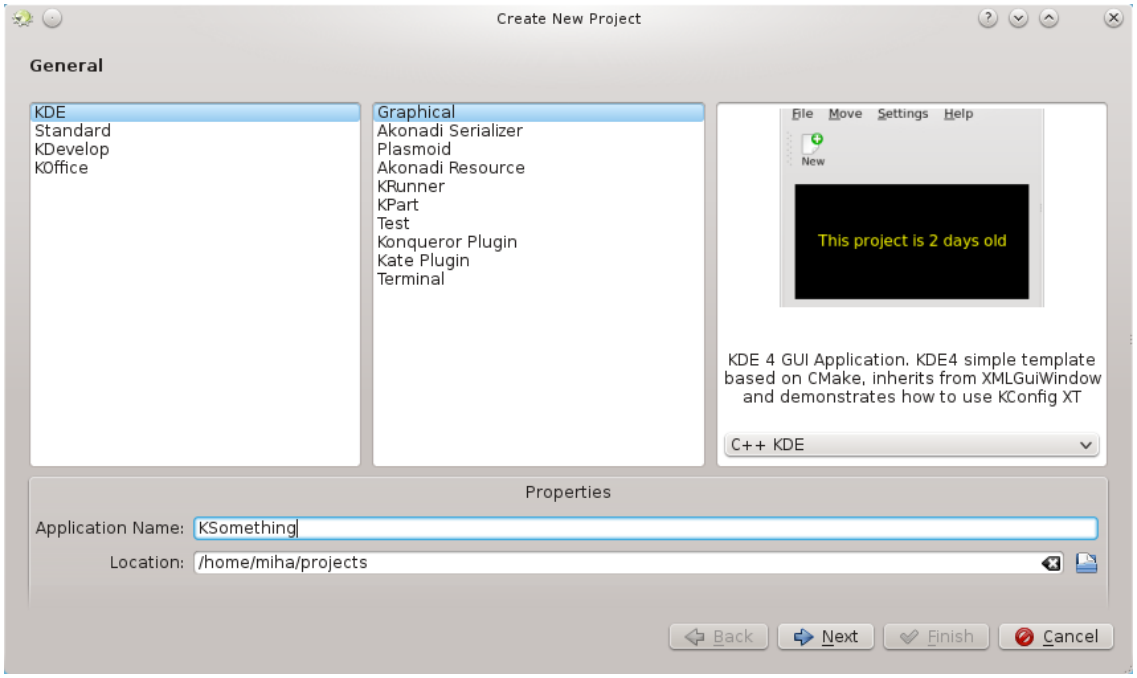
2.3 Configurar una aplicació com un segon projecte

La propera cosa que voldreu fer és crear altres projectes en la mateixa sessió. En l'exemple anterior, voleu afegir el controlador de dispositiu com el segon projecte, el qual es pot fer emprant exactament els mateixos passos.

Si teniu múltiples aplicacions o biblioteques, simplement repetiu els passos per a afegir més i més projectes a la vostra sessió.

2.4 Crear projectes a partir de zero

Per descomptat també hi ha la possibilitat d'iniciar un projecte nou des de zero. Això es pot fer emprant l'element de menú **Projectes** → **Nou des d'una plantilla...**, el qual presentarà un diàleg per a la selecció de plantilles. Algunes plantilles de projectes venen amb el KDevelop, però encara n'hi ha més de disponibles mitjançant la instal·lació de l'aplicació KAppTemplate. Escolliu el tipus de projecte i el llenguatge de programació des del diàleg, introduïu un nom i una ubicació pel vostre projecte, i feu clic a **Següent**.



La segona pàgina del diàleg permet establir un sistema pel control de versions. Trieu el sistema que voleu utilitzar, i completeu si cal la configuració del sistema. Si no voleu emprar un sistema pel control de versions, o el voleu configurar manualment després, escolliu **Cap**. Una vegada satisfet amb la vostra elecció, premeu **Finalitza**.

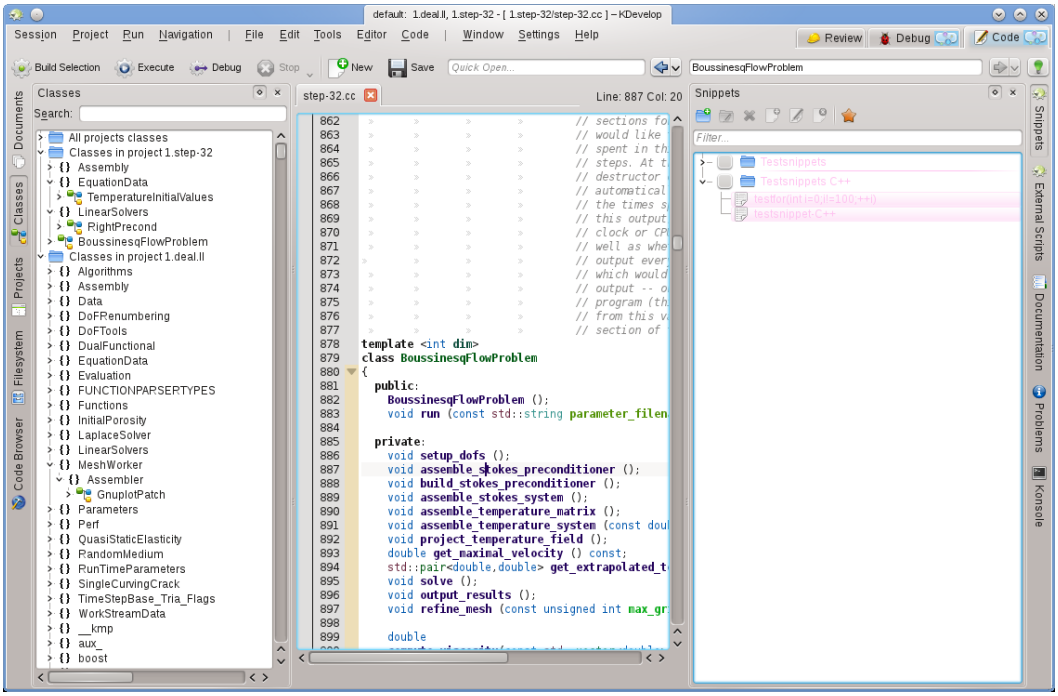
El vostre projecte està creat, de manera que podeu provar la construcció o instal·lació. Algunes plantilles inclouran comentaris dins del codi, o fins i tot un fitxer README a part, i es recomana que primer els llegiu. A continuació, podreu començar a treballar en el vostre projecte, afegint-li les característiques que vulgueu.

Capítol 3

Treballar amb el codi font

A més de la depuració, llegint i escrivint codi font és com passareu la major part del temps durant el desenvolupament de programari. Amb aquesta finalitat, el KDevelop ofereix moltes maneres diferents d'explorar el codi font i per a fer que l'escriptura sigui més productiva. Tot plegat s'analitza en més detall en les seccions següents, el KDevelop no és només un editor de codi font -més aviat, és un sistema de gestió de les fonts que us dona diferents punts de vista de la informació extreta dels fitxers que en conjunt formen el codi font de la vostra sessió-.

3.1 Eines i vistes



Per tal de treballar amb els projectes, el KDevelop té el concepte d'*eines*. Una eina proporciona una visió particular de la font, o una acció que es pot dur a terme amb aquesta. Les eines estan representades per botons al voltant de tot el perímetre de la finestra (amb text en vertical al llarg dels marges esquerre i dret, o en horitzontal al llarg del marge inferior). Si feu clic sobre un d'ells,

s'expandirà a una subfinestra *-vista-* dins de la finestra principal. Si feu clic al botó de l'eina de nou, la subfinestra desapareixerà.

Per a fer que una subfinestra desaparegui, també podeu fer clic sobre la **x** a la part superior dreta de la subfinestra.

La imatge de dalt mostra una selecció particular d'eines, alineades al marge esquerre i dret; a la imatge, l'eina **Classes** està oberta a l'esquerra i l'eina **Retalls** a la dreta, juntament amb un editor per a un fitxer de codi font al centre. A la pràctica, la major part del temps és probable que només tingueu l'editor i potser l'eina **Classes** o **Navegador de codi** oberta a l'esquerra. Una altra vista d'eina és probable que només s'obri temporalment quan s'utilitza aquesta eina, deixant més espai per a l'editor la major part del temps.

En executar el KDevelop la primera vegada, ja hauríeu de tenir el botó de l'eina **Projectes**. Feu clic sobre seu: s'obrirà una subfinestra mostrant a la part inferior els projectes que heu afegit a la sessió, i una vista a la part superior del sistema de fitxers dels directoris dels vostres projectes.

Hi ha moltes altres eines que podeu utilitzar amb el KDevelop, no totes són inicialment presents com a botons en el perímetre. Per a afegir alguna cosa, aneu a l'entrada del menú **Finestres** → **Afegeix una vista d'eina**. Aquestes són algunes de les que probablement trobeu d'utilitat:

- **Classes**: Una llista completa de totes les classes que estan definides en un dels projectes o la vostra sessió amb la totalitat de les seves funcions de membre i variables. En fer clic sobre qualsevol dels membres s'obrirà una finestra de l'editor de codi font en la ubicació de dit l'element.
- **Documents**: Una llista d'alguns dels fitxers visitats més recentment, per tipus (p. ex., fitxers de codi font, fitxers de pedaç, documents de text pla).
- **Navegador de codi**: En funció de la posició del cursor en un fitxer, aquesta eina mostrarà les coses que hi estan relacionades. Per exemple, si us trobeu a una línia `#include`, mostrarà informació sobre el fitxer que està inclòs així com quines classes es declaren en aquest fitxer; si us trobeu en una línia buida a l'àmbit del fitxer, mostrarà les classes i funcions declarades i definides en el fitxer actual (tot com enllaços: fent clic sobre seu anireu al punt en el fitxer on es fa la declaració o definició real); si us trobeu en una definició de funció, mostrarà on es troba la declaració i oferirà una llista dels llocs on s'empra dita funció.
- **Sistema de fitxers**: Mostra una vista en arbre del sistema de fitxers.
- **Documentació**: Permet cercar a les pàgines de manual i altres documents d'ajuda.
- **Retalls**: Proporciona seqüències de text que s'empren sovint i no es volen escriure cada vegada. Per exemple, en el projecte des del qual es va crear la imatge de dalt, hi ha una freqüent necessitat d'escriure codi com

```
for (typename Triangulation< dim>::active_cell_iterator cell
      = triangulation.begin_active();
   cell != triangulation.end();
   ++cell)
```

Aquesta és una expressió peculiar, però apareixerà quasi exactament com aquesta cada vegada que necessiteu un bucle com aquest -el qual el faria un bon candidat per a un retall-.

- **Konsole**: Obre una finestra de la línia d'ordres a l'interior de la finestra principal del KDevelop, per a una ordre ocasional que possiblement voleu introduir (p. ex., per a executar `./configure`).

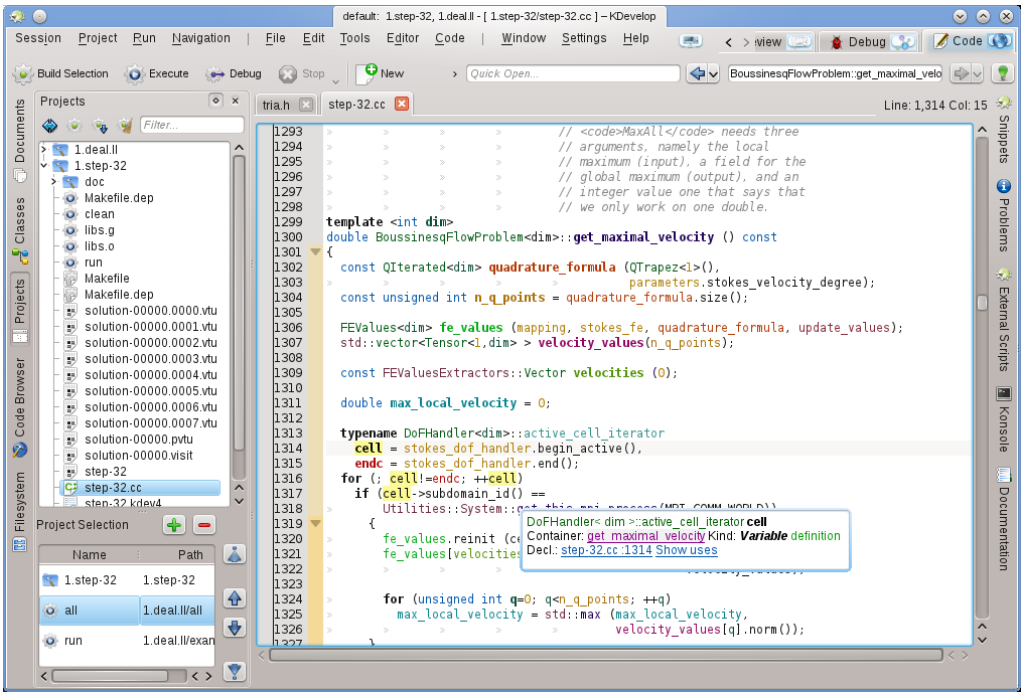
Una llista completa de les eines i vistes es troba aquí.

Per a molts programadors, l'espai vertical de la pantalla és el més important. Amb aquesta finalitat, podeu organitzar les vostres vistes d'eina al marge esquerre i dret de la finestra: per a moure una eina, feu clic en el seu símbol amb el botó dret del ratolí i seleccioneu una posició nova.

3.2 Explorar el codi font

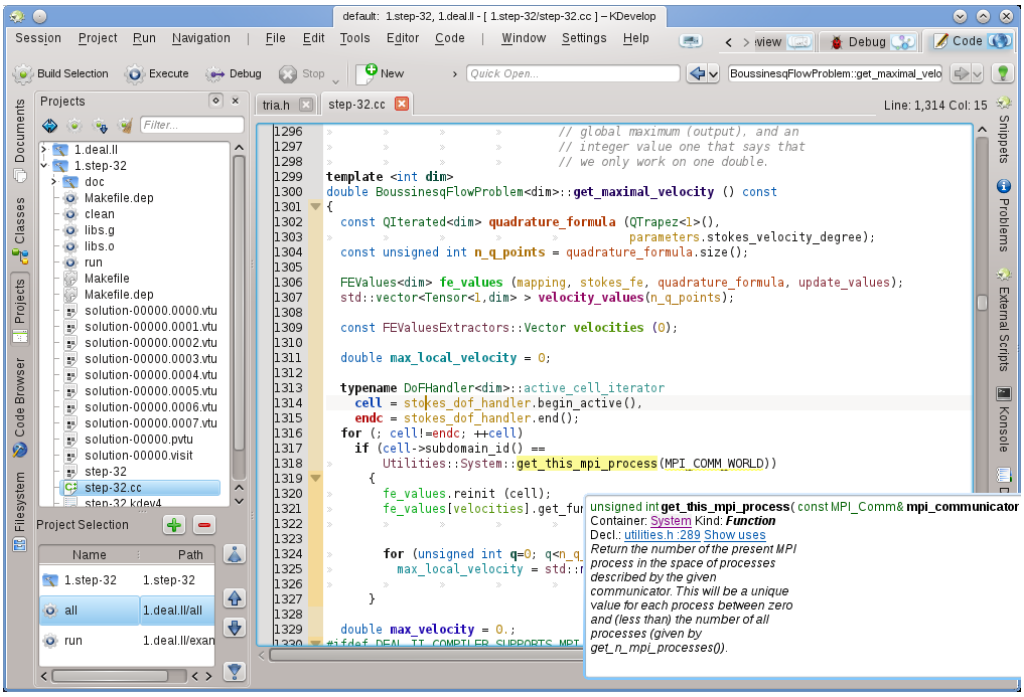
3.2.1 Informació local

El KDevelop *entén* el codi font, i en conseqüència és molt bo proporcionant informació sobre les variables o funcions que puguin aparèixer en el vostre programa. Per exemple, aquí teniu una captura de pantalla del treball amb un tros de codi i passar el ratolí sobre el símbol `cel·la` a la línia 1316 (si esteu treballant orientat amb el teclat, podeu obtenir el mateix efecte prement la tecla **Alt** durant un temps):



El KDevelop mostra un text d'ajuda que inclou el tipus de la variable (en aquest cas: `DoFHandler <dim>active_cell_iterator`), on es declara aquesta variable (el *contenidor*, que aquí és la funció que l'envolta `get_maximal_velocity` perquè és una variable local), el qual és (una variable, no una funció, classe o espai de noms) i on es declara (a la línia 1314, només unes poques línies amunt en el codi).

En el context actual, el símbol sobre el qual es mou el ratolí no té cap documentació associada. En aquest exemple, el ratolí planava sobre el símbol `get_this_mpi_process` a la línia 1318, el resultat serà el següent:

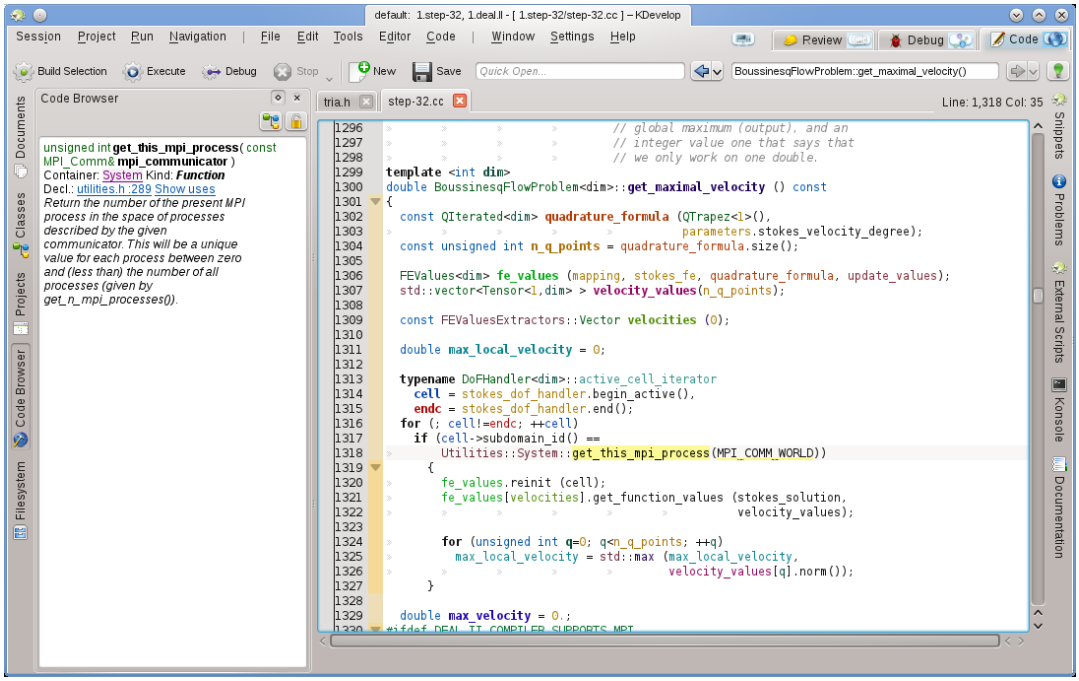


Aquí, el KDevelop té una referència creuada de la declaració des d'un fitxer completament diferent (`utilities.h`, el qual de fet resideix en un projecte diferent de la mateixa sessió) juntament amb el comentari a l'estil doxygen que acompanya la declaració.

El que fa aquests consells d'eina encara més útils és que són dinàmics: puc fer clic al contenidor per a obtenir informació sobre el context en el qual es declara la variable (és a dir, sobre l'espai de noms `System`, com on es declara, defineix, s'utilitza, o on es troba la seva documentació) i jo puc fer clic als enllaços blaus per a reiniciar la posició del cursor a la ubicació de la declaració del símbol (p. ex., en `utilities.h`, línia 289) o donar-me una llista dels llocs on s'utilitza aquest símbol al fitxer actual o en tot tots els projectes de la sessió actual. Això últim sovint és útil si voleu explorar com, p. ex., s'empra una funció en particular en una gran base de codi.

NOTA

La informació continguda en un consell d'eina és efímera -doncs dependrà de mantenir premuda la tecla **Alt** o que hi feu passar el ratolí-. Si voleu un lloc més permanent, obriu la vista d'eina **Navegador de codi** en una de les subfinestres. Per exemple, aquí el cursor es troba sobre la mateixa funció que en l'exemple anterior, i la vista d'eina de l'esquerra presenta el mateix tipus d'informació com en el consell d'eina d'abans:



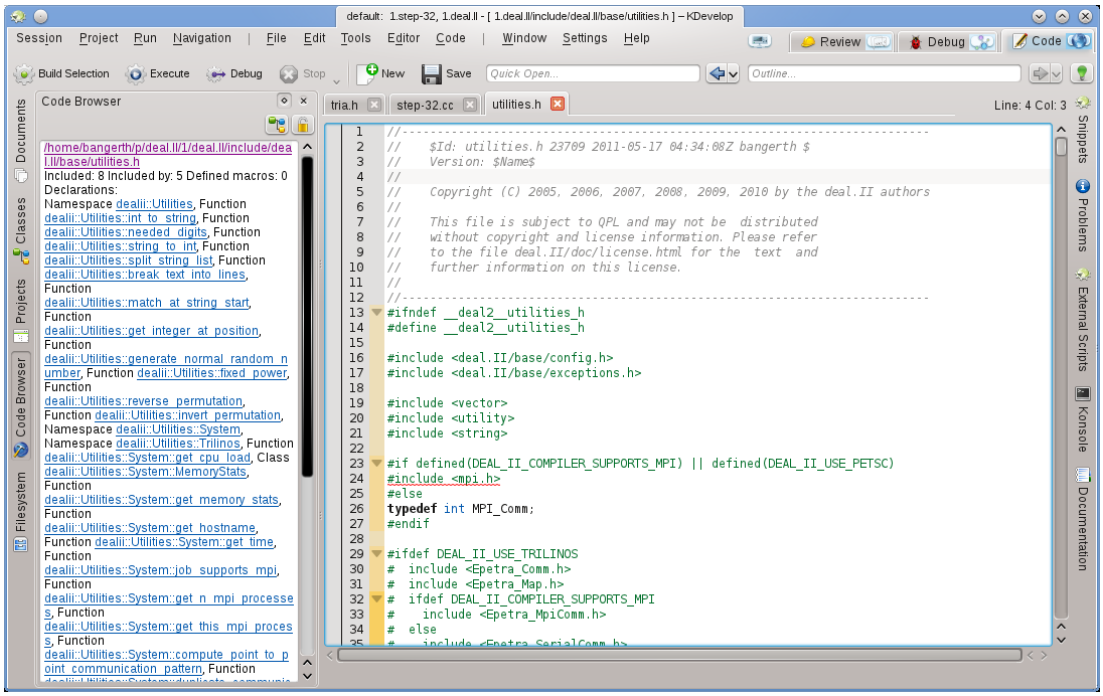
En moure el cursor cap a la dreta es canviarà la informació presentada cap a l'esquerra. El que és més, en fer clic al botó **Bloqueja la vista actual** a la part superior dreta us permetrà bloquejar aquesta informació, fent-la independent del moviment del cursor mentre exploreu la informació que us presenta.

NOTA

Aquest tipus d'informació contextual es troba disponible en molts altres llocs al KDevelop, no només a l'editor del codi font. Per exemple, mantenint premuda la tecla **Alt** en una llista de compleció (p. ex., quan feu una obertura ràpida) també produirà informació contextual sobre el símbol actual.

3.2.2 Informació sobre l'àmbit dels fitxers

El següent nivell és obtenir informació sobre el fitxer font sobre el qual esteu treballant. Per a fer-ho, situeu el cursor a l'àmbit del fitxer al fitxer actual i cerqueu el que mostra la vista d'eina **Navegador de codi**:

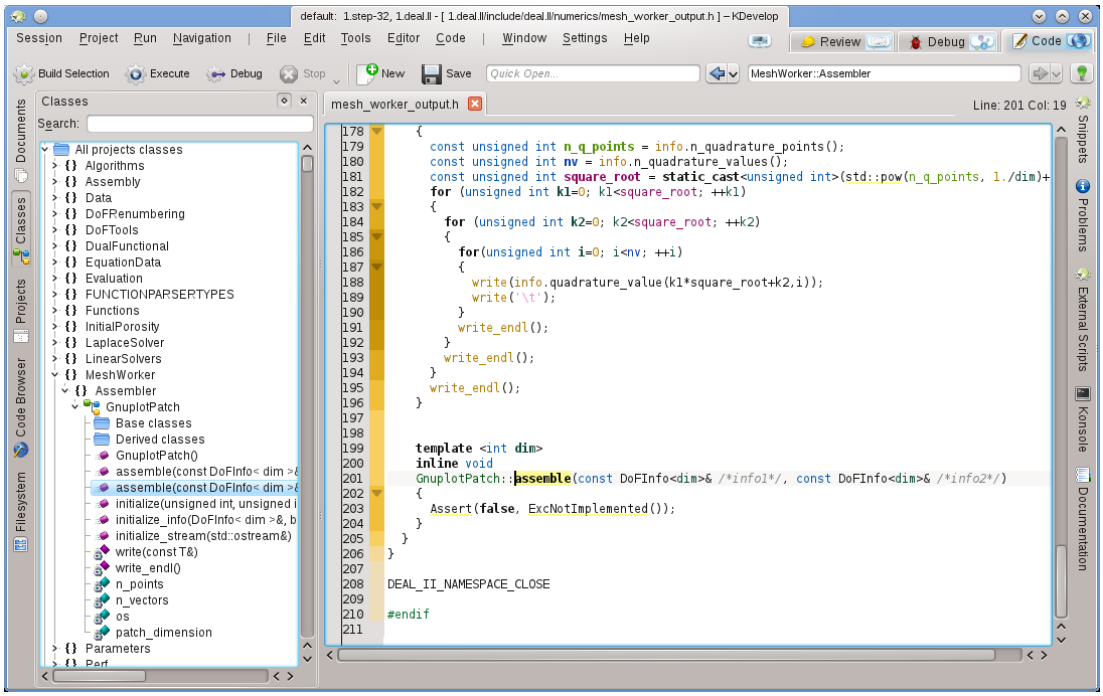


Aquí, es mostra una llista dels espais de noms, classes i funcions declarats o definits al fitxer actual, el qual us dona una visió general del que succeeix en aquest fitxer i un mitjà per a saltar directament a qualsevol d'aquestes declaracions o definicions sense haver de desplaçar-vos cap amunt i avall o a la cerca d'un símbol en particular.

> NOTA
> La informació que es mostra per a l'àmbit del fitxer és la mateixa que es presenta en el mode «Esquema» discutit a continuació en navegar pel codi font; la diferència és que el mode d'esquema només és un consell d'eina temporal.

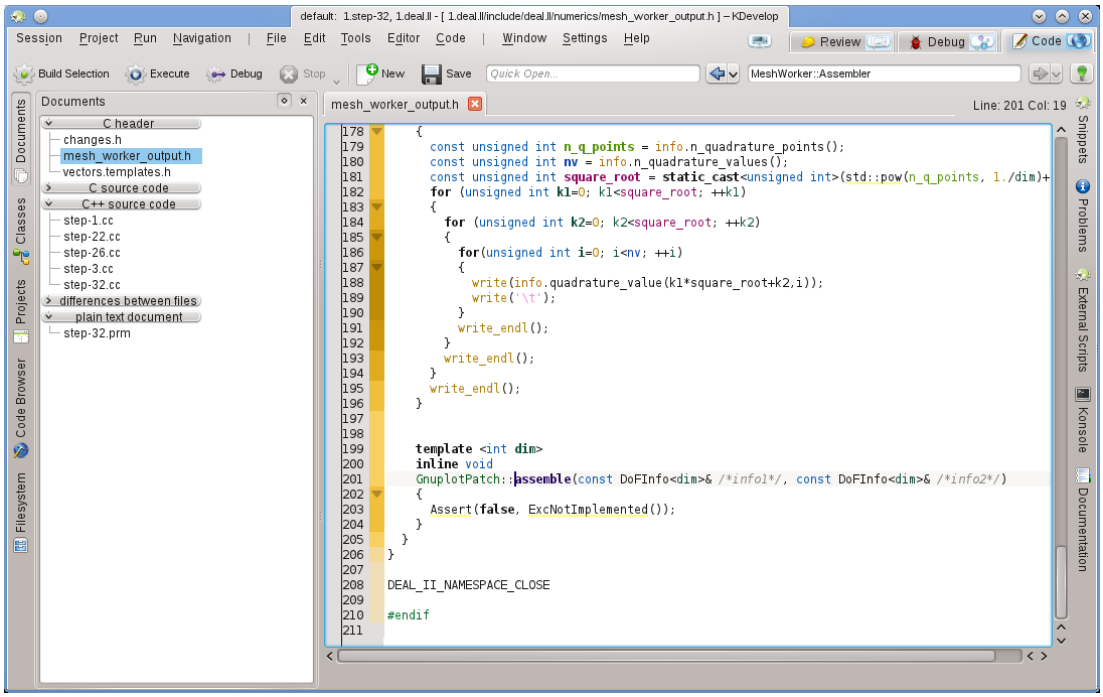
3.2.3 Informació sobre l'àmbit del projecte i la sessió

Hi ha moltes maneres d'obtenir informació sobre tot un projecte (o, de fet, quant a tots els projectes en una sessió). Aquest tipus d'informació es proporciona normalment a través de diferents vistes d'eina. Per exemple, la vista d'eina **Classes** ofereix una estructura en arbre de totes les classes i espais de nom dels voltants per a tots els projectes en una sessió, juntament amb les funcions de membre i variables de cadascuna d'aquestes classes:



En passar el ratolí sobre una entrada ofereix de nou la informació sobre el símbol, la seva ubicació i definició de la declaració, i els seus usos. En fer doble clic sobre una entrada en aquesta vista en arbre s'obrirà una finestra d'edició en el lloc on es declara o defineix aquest símbol.

Però hi ha altres maneres de veure la informació global. Per exemple, l'eina **Documents** proporciona una vista d'un projecte en termes de les classes de fitxers o altres documents que conté aquest projecte:



3.2.4 Explicació del ressaltat amb els colors de l'Arc de Sant Martí

El KDevelop empra una varietat de colors per a ressaltar diferents objectes en el codi font. Si sabeu el que signifiquen els diferents colors, podeu extreure ràpidament una gran quantitat d'informació a partir del codi font amb només mirar els colors, sense llegir un sol caràcter. Les regles del ressaltat són les següents:

- Els objectes del tipus «class» / «struct» i «enum» (els valors i el tipus), les funcions (globals), i els membres de la classe tenen cadascun el seu propi color assignat (les classes en verd, les enumeracions en vermell fosc, i els membres en groc o violeta fosc, les funcions (globals) sempre en violeta).
- Totes les variables globals es mostren de color verd fosc.
- Els identificadors que són «typedefs» per a altres tipus es mostren en verd blavós.
- Totes les declaracions i definicions dels objectes estan en negreta.
- Si s'accedeix a un membre des de dins del context en el qual es defineix (base o classe derivada) aquest apareixerà en groc, en cas contrari, apareixerà en violeta.
- Si un membre és privat o protegit, es mostrarà amb un color lleugerament més fosc quan s'utilitza.
- Per a les variables locals en l'àmbit del cos d'una funció es trien els colors de l'arc de Sant Martí basant-se en un codi numèric de l'identificador. Això inclou els paràmetres de la funció. Un identificador sempre tindrà el mateix color dins del seu àmbit (però el mateix identificador tindrà un color diferent si representa un objecte diferent, és a dir, si es torna a definir en un àmbit més niat), i en general obtindreu el mateix color per al mateix nom d'identificador en àmbits diferents. Per tant, si teniu múltiples funcions que prenen els paràmetres amb els mateixos noms, els arguments tots seran del mateix color. Aquests colors de l'arc de Sant Martí es poden desactivar per separat de la coloració global en el diàleg de configuració.
- Els identificadors per als quals el KDevelop no pugui determinar la declaració corresponent, seran de color blanc. Això de vegades pot ser causat per la falta de les directives `#include`.
- A més d'aquesta coloració, s'aplicarà el ressaltat de sintaxi normal de l'editor, com es coneix a partir del ressaltat semàntic del Kate. Si hi ha un conflicte, el KDevelop sempre anul·larà el ressaltat de l'editor.

3.3 Navegar pel codi font

A la secció anterior, hem debatut l'exploració del codi font, és a dir, l'obtenció d'informació sobre els símbols, fitxers i projectes. El següent pas és voltar per la vostra base de codi font, és a dir, navegar-hi. En aquest cas hi ha diferents nivells en què això és possible: local, dins d'un fitxer, i dins d'un projecte.

> NOTA
> Podeu accedir a moltes de les formes de navegar a través del codi des del menú **Navegació** a la finestra principal del KDevelop.

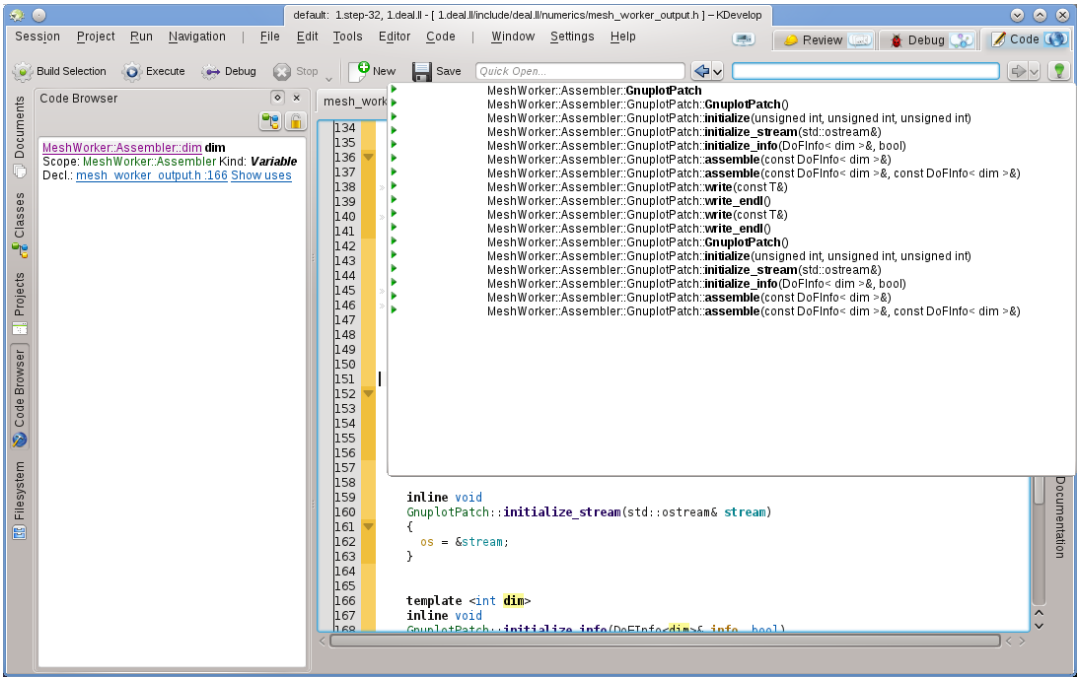
3.3.1 Navegació local

El KDevelop és molt més que un editor, *també* és un editor de codi font. Com a tal, per descomptat podeu moure el cursor cap amunt, avall, esquerra o dreta en un fitxer de codi font. També podeu utilitzar les tecles **Re Pàg** i **Av Pàg**, i totes les altres ordres que s'utilitzen des de qualsevol editor d'utilitat.

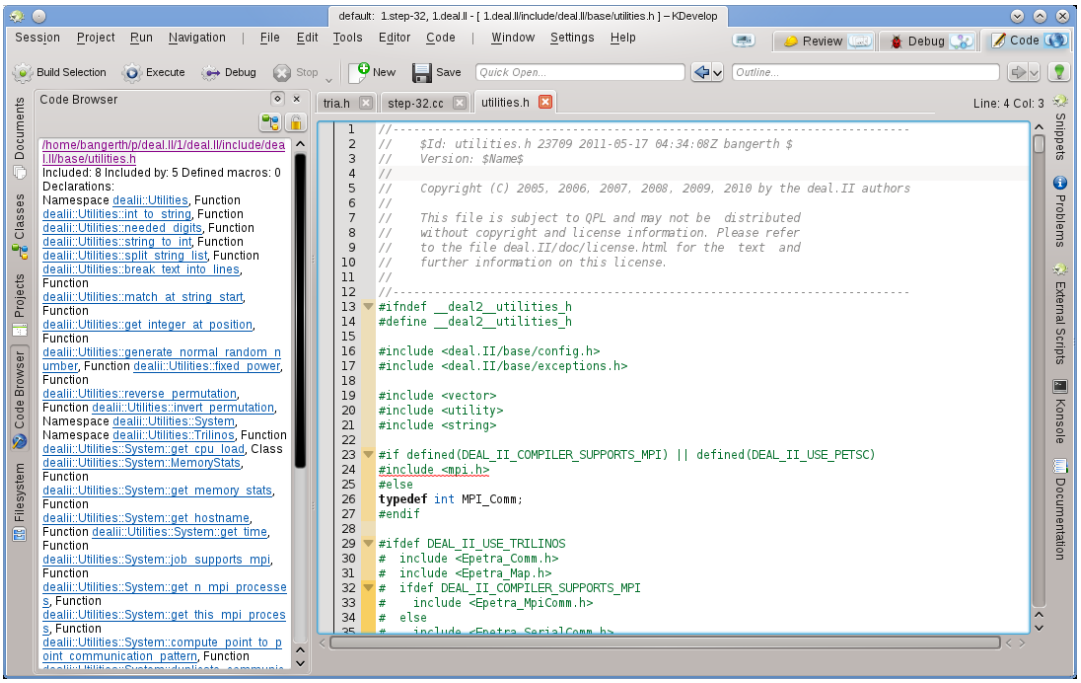
3.3.2 Navegació en l'àmbit de fitxers i mode d'esquema

En l'àmbit del fitxer, el KDevelop ofereix moltes possibles maneres per a navegar a través del codi font. Per exemple:

- **Esquema**: Podeu obtenir un esquema de què hi ha al fitxer actual en almenys tres formes diferents:
  - En fer clic al quadre de text **Esquema** a la part superior dreta de la finestra principal, o prement **Alt-Ctrl-N** s'obrirà un menú desplegable mostrant totes les declaracions de funció i de classe:



    A continuació, podeu simplement seleccionar-ne una a la qual anar, o -si n'hi ha un munt- començar a escriure qualsevol text que pot aparèixer en els noms que es mostren; en aquest cas, a mesura que escriviu, la llista es farà més i més petita a mesura que s'eliminen els noms que no coincideixen amb el text ja escrit, fins que esteu preparat per a seleccionar una de les opcions.
  - Posicionar el cursor en l'àmbit del fitxer (és a dir, fora de qualsevol declaració o definició de funció o de classe) i tenir oberta l'eina **Explorador de codi**:



Això també us proporciona un esquema de què està succeint al fitxer actual, i us permet seleccionar on voleu saltar.

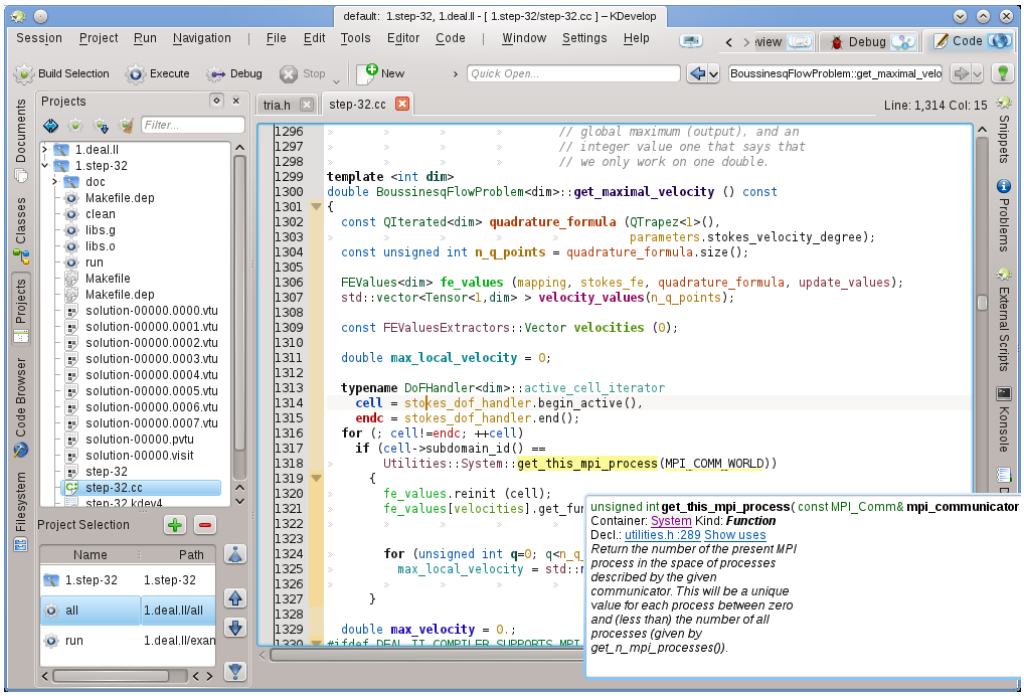
– En passar el punter del ratolí sobre la pestanya d'un dels fitxers oberts també es mostra un esquema del fitxer en aquesta pestanya.

- Els fitxers de codi font s'organitzen com una llista de declaracions o definicions de funcions. En prémer **Alt-Ctrl-Re Pàg** i **Alt-Ctrl-Av Pàg** saltareu a la definició de la funció anterior o següent en aquest fitxer.

3.3.3 Navegar sobre l'àmbit del projecte i la sessió: Navegació semàntica

Com s'ha esmentat en altres llocs, el KDevelop, en general no considera els fitxers de codi font de forma individual, si no més aviat té en compte els projectes com un tot (o, més aviat, tots els projectes que formen part de la sessió actual). Com a conseqüència, ofereix moltes possibilitats per a navegar a través de tots els projectes. Algunes es deriven del que ja hem discutit en l'apartat Explorar el codi font, mentre que altres són realment diferents. El punt en comú és que aquestes característiques de navegació es basen en una *comprensió semàntica* del codi, és a dir, us ofereix quelcom que requereix analitzar per complet els projectes i connectar les dades. La següent llista mostra algunes maneres de com navegar a través del codi font, el qual pot restar dispers al llarg d'un nombre potencialment molt gran de fitxers:

- Com s'ha vist en l'apartat Explorar el codi font, podeu obtenir consells amb informació sobre espai de noms, classes, funcions o els noms de les variables, amb només passar el ratolí sobre seu o mantenint premuda la tecla **Alt** durant un temps. Heus aquí un exemple:



En fer clic als enllaços per a la declaració d'un símbol o desplegar la llista d'usos podreu saltar a aquestes ubicacions, si cal, obrirà el fitxer corresponent i col·locarà el cursor a la posició corresponent. Es pot aconseguir un efecte similar emprant la vista d'eina **Navegador de codi**, com ja se n'ha parlat anteriorment.

- Una manera més ràpida d'arribar a la declaració d'un símbol sense haver de fer clic sobre els enllaços del consell d'eina és habilitar temporalment **Mode de navegació del codi font** prement la tecla **Alt** o **Ctrl**. En aquest mode, és possible fer clic directament sobre qualsevol símbol de l'editor per a passar a la seva declaració.

- **Obertura ràpida**: Una manera molt poderosa de saltar a altres fitxers o ubicacions és emprar els diferents mètodes d'*obertura ràpida* al KDevelop. Hi ha quatre versions d'aquesta:

  – **Obre ràpidament una classe** (**Navegació** → **Obre ràpidament una classe** o **Alt-Ctrl-C**): Obtindreu una llista de totes les classes en aquesta sessió. Comenceu a escriure (una part) el nom d'una classe i la llista es reduirà gradualment a només aquells que realment coincideixin amb el que heu escrit. Si la llista és prou curta, seleccioneu un element emprant les tecles amunt i avall, i el KDevelop us durà al lloc on es declara la classe.

  – **Obre ràpidament una funció** (**Navegació** → **Obre ràpidament una funció** o **Alt-Ctrl-M**): Obtindreu una llista de totes (membres) les funcions que formen part dels projectes a la sessió actual, i podreu seleccionar-ne una de la mateixa manera com abans. Tingueu en compte que aquesta llista pot incloure tant declaracions com definicions de funcions.

  – **Obre ràpidament un fitxer** (**Navegació** → **Obre ràpidament un fitxer** o **Alt-Ctrl-O**): Obtindreu una llista de tots els fitxers que formen part dels projectes a la sessió actual, i podreu seleccionar-ne un de la mateixa manera que abans.

  – **Obertura ràpida universal** (**Navegació** → **Obertura ràpida** o **Alt-Ctrl-Q**): Si oblideu quina és la combinació de tecles adequada per a les ordres anteriors, aquesta és la navalla suïssa universal -simplement us mostrarà una llista combinada de tots els fitxers, funcions, classes i altres coses que podeu seleccionar-.

- **Salta a la declaració/definició**: Quan s'implementa una funció (membre), sovint és necessari tornar al punt on es declara una funció, p. ex., per a mantenir la llista d'arguments de la funció sincronitzats entre la declaració i la definició, o per a actualitzar la documentació. Per a fer-ho, situeu el cursor sobre el nom de la funció i seleccioneu **Navegació** → **Salta a la declaració** (o premeu **Ctrl-.**) per a arribar al lloc on es declara la funció. Hi ha múltiples maneres de tornar al lloc original:

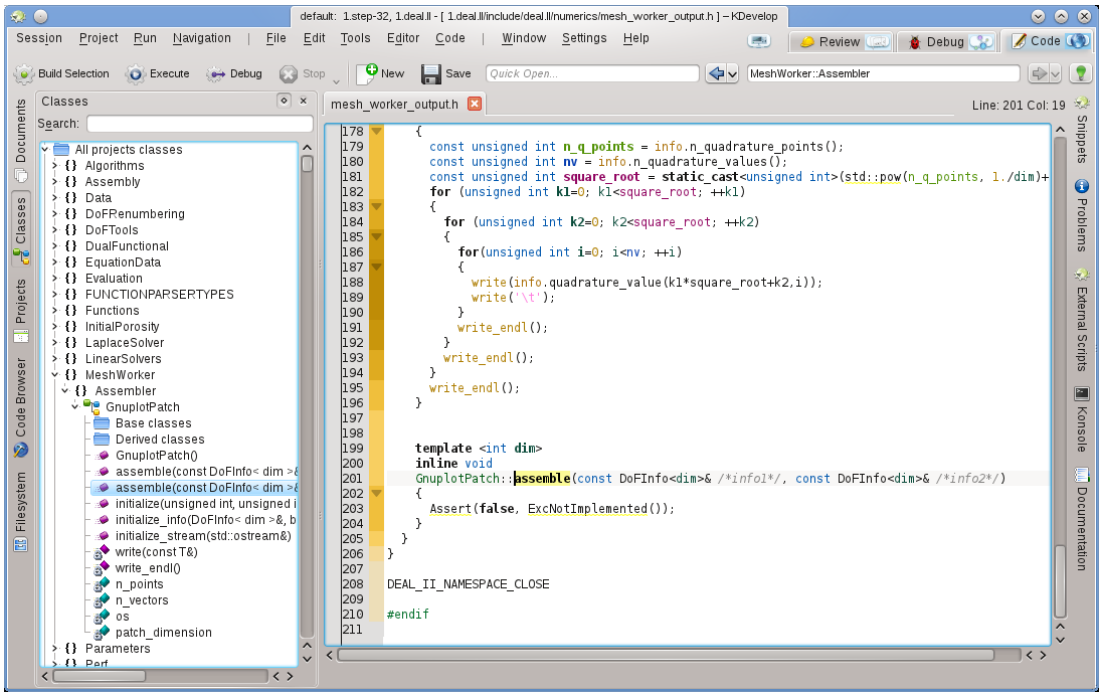
- Seleccionant **Navegació** → **Salta a la definició** (o prement **Ctrl-,**).
- Seleccionant **Navegació** → **Context visitat anterior** (o prement **Meta-Fletxa esquerra**), com es descriu a continuació.

NOTA
El fet de saltar a la declaració d'un símbol no només funciona quan se situa el cursor sobre el nom de la funció que esteu implementant. També funciona amb altres símbols: Situant el cursor sobre una variable (local, global o membre) i saltarà a la ubicació on es declara. De manera similar, podeu situar el cursor sobre el nom d'una classe, p. ex., en una variable de declaració d'una funció, i saltar a la ubicació on es declara.

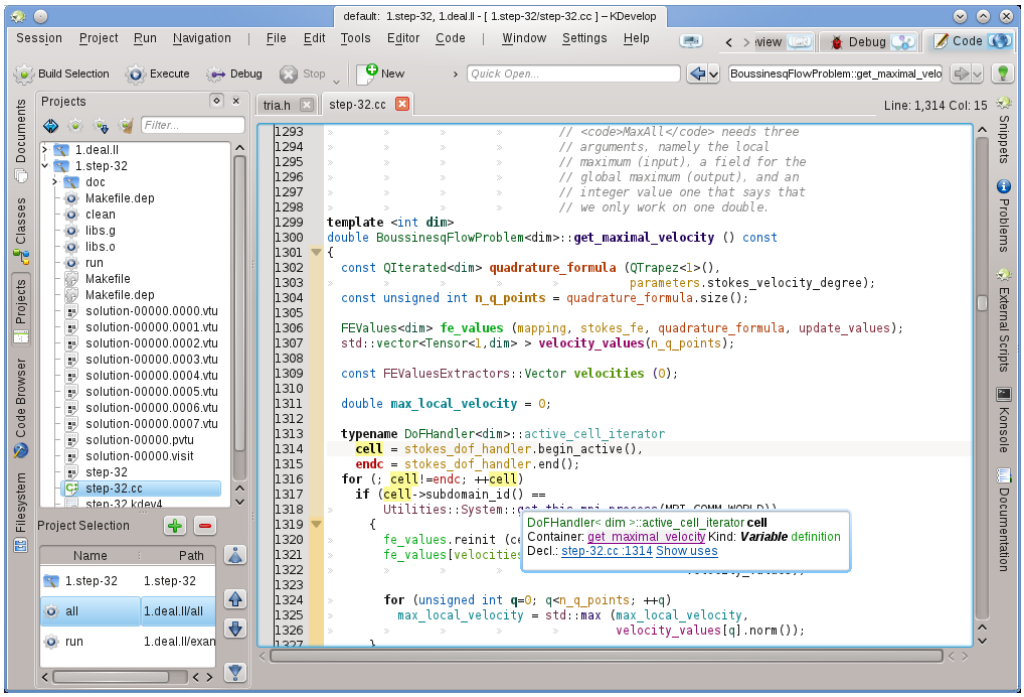
- **Canvia entre la declaració/definició**: En l'exemple anterior, per a saltar a la ubicació de la declaració de la funció actual, primer és necessari situar el cursor sobre el nom de la funció. Per a evitar aquest pas, podeu seleccionar **Navegació** → **Canvia entre la declaració/definició** (o prémer **Maj-Ctrl-C**) per a saltar a la declaració de la funció dins la qual es troba actualment el cursor. Seleccionant la mateixa entrada de menú una segona vegada tornareu a la ubicació on es defineix la funció.
- **Ús anterior/següent**: En situar el cursor sobre el nom d'una variable local i seleccionant **Navegació** → **Ús següent** (o prement **Meta-Maj-Fletxa dreta**) anireu a l'ús següent d'aquesta variable en el codi. (Tingueu en compte que això no es limita a cercar la següent ocurrència del nom de la variable, si no més aviat té en compte que les variables amb el mateix nom però en diferents àmbits són diferents). El mateix funciona per a l'ús dels noms de la funció. Seleccionant **Navegació** → **Ús anterior** (o prement **Meta-Maj-Fletxa esquerra**) anireu a l'ús anterior d'un símbol.

NOTA
Per a veure la llista de tots els usos d'un nom a través del qual fan un cicle aquestes ordres, poseu el cursor sobre seu i obriu la vista d'eina **Navegador de codi** o premeu i mantingueu premut el botó **Alt**. Això s'explica en més detall a la secció sobre Explorar el codi.

- La **llista contextual**: Els navegadors web tenen la característica on podeu anar cap enrere i cap endavant en la llista de pàgines web visitades més recents. El KDevelop té el mateix tipus de característiques, excepte que en lloc de pàgines web, visiteu *contextos*. Un context és la posició actual del cursor, i podeu canviar-lo navegant fora seu emprant qualsevol mètode excepte ordres del cursor -p. ex., fent clic en un lloc proporcionat per un consell, a la vista d'eina del **Navegador de codi**, una de les opcions donades al menú **Navegació**, o qualsevol altre ordre de navegació-. Emprant **Navegació** → **Context visitat anterior** (**Meta-Fletxa esquerra**) i **Navegació** → **Context visitat següent** (**Meta-Fletxa dreta**) anireu al llarg d'aquesta llista de contexts visitats de la mateixa manera com els botons **enrere** i **endavant** d'un navegador us porten a la pàgina web anterior i següent a la llista de les pàgines visitades.
- Finalment, hi ha vistes d'eina que us permeten navegar a diferents llocs a la base del codi. Per exemple, l'eina **Classes** proporciona una llista de tots els espais de noms i classes en tots els projectes de la sessió actual, i permet expandir els seus elements per les funcions de membre i variables de cadascuna d'aquestes classes:



En fer doble clic sobre un element (o anar a través del menú contextual amb el botó dret del ratolí) saltareu a la ubicació on es declara aquest element. Altres eines permeten coses similars; p. ex., la vista d'eina **Projectes** proporciona una llista dels fitxers que formen part d'una sessió:



Una vegada més, fent doble clic en un fitxer l'obrireu.

3.4 Escriure codi font

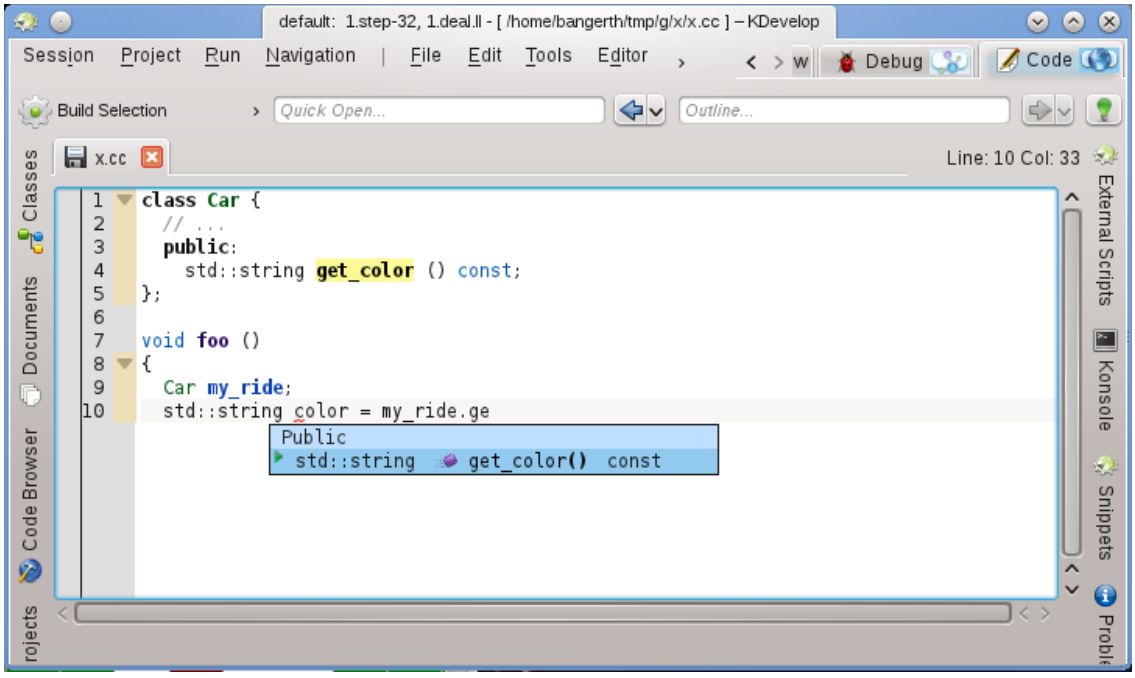
Atès que el KDevelop entén el codi font dels vostres projectes, us pot ajudar a escriure més codi. A continuació, es descriuen algunes de les formes en les quals fa això.

3.4.1 Compleció automàtica

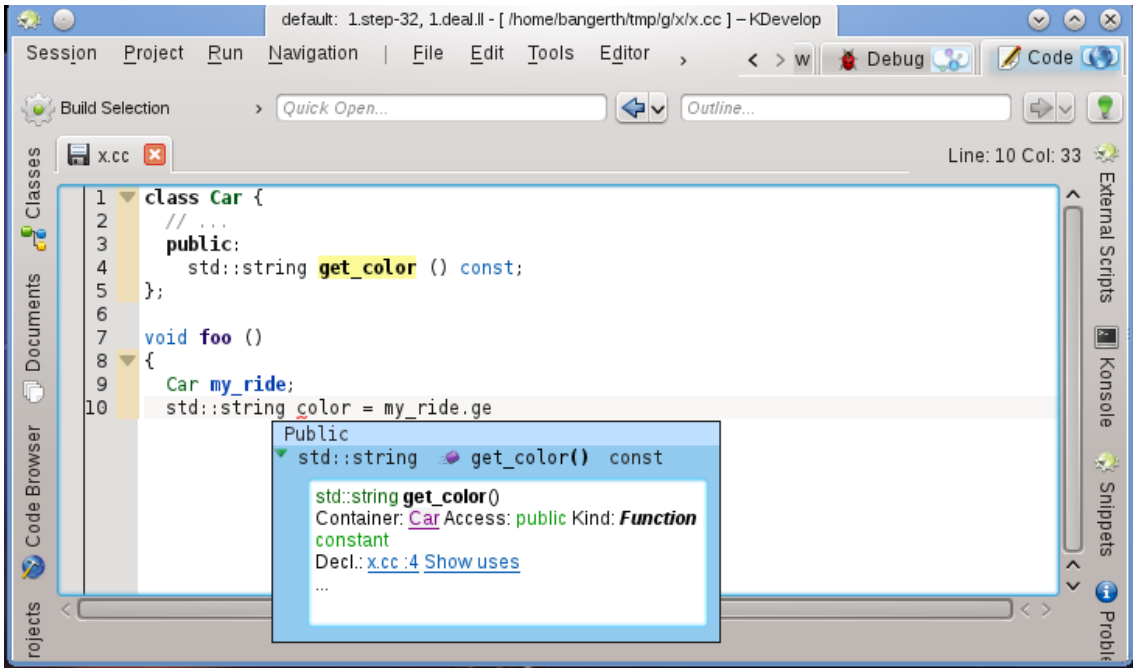
Probablement la més útil de totes les característiques en escriure codi nou és la compleció automàtica. Considerem, p. ex., el següent fragment de codi:

```
class Car {
  // ...
  public:
    std::string get_color () const;
};
void foo()
{
  Car my_ride;
  // ...do something with this variable...
  std::string color = my_ride.ge
```

En l'última línia, el KDevelop recordarà que la variable `my_ride` és del tipus `Car`, i automàticament oferirà completar el nom de la funció de membre `ge` com `get_color`. De fet, tot el que haureu de fer és seguir escrivint fins que la funció de compleció automàtica hagi reduït el nombre de coincidències a una, i després prémer la tecla **Retorn**:



Tingueu en compte que podreu fer clic al consell per a obtenir més informació sobre la funció, a part del seu tipus de retorn i si és pública:



La compleció automàtica us pot estalviar un munt d'escriure si el vostre projecte utilitza noms llargs per a les variables i funcions; a més, evita errors d'ortografia (i els errors del compilador resultants) i fa que sigui molt més fàcil de recordar els noms exactes de les funcions; p. ex., si tots els vostres «get» comencen amb `get_`, la característica de compleció automàtica només us mostrarà una llista de possibles coincidències quan hàgiu escrit les primeres quatre lletres, és probable que durant el procés us recordi la funció correcta. Tingueu en compte que perquè funcioni la compleció automàtica, ni la declaració de la classe `Car` ni de la variable `my_ride` necessiten estar en el mateix fitxer on actualment esteu escrivint codi. El KDevelop simplement ha de saber que aquestes classes i variables estan connectades, és a dir, els fitxers en els quals es realitzen aquestes connexions han de formar part del projecte on esteu treballant.

NOTA

El KDevelop no sempre sap quan ajudar amb la compleció del codi. Si el consell de compleció automàtica no s'obre automàticament, premeu **Ctrl-Espai** per a obrir manualment una llista de complecions. En general, per tal que funcioni la compleció automàtica, el KDevelop necessita analitzar els fitxers d'origen. Això succeeix en segon pla per a tots els fitxers que formen part dels projectes a la sessió actual després d'iniciar el KDevelop, així com després de deixar d'escriure per una fracció de segon (el retard es pot configurar).

NOTA

El KDevelop només analitzarà els fitxers que consideri codi font, segons el que determina el tipus MIME del fitxer. Aquest tipus no s'estableix abans de la primera vegada que es desa un fitxer; en conseqüència, la creació d'un fitxer nou i començar a escriure codi no donarà lloc a analitzar per a la compleció automàtica fins després que es desi per primera vegada.

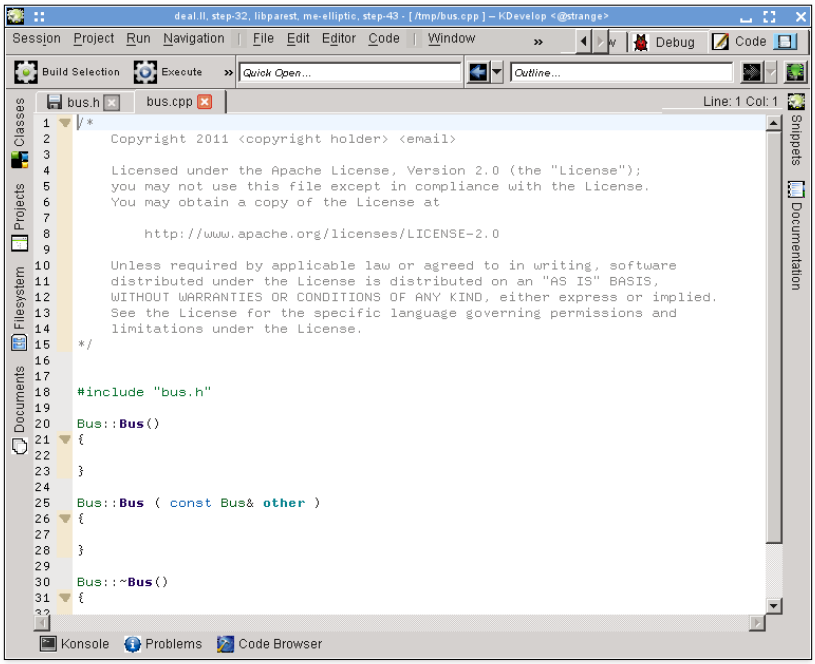
NOTA
Com en la nota anterior, perquè funcioni la compleció automàtica, el KDevelop ha de ser capaç de trobar les declaracions en els fitxers de capçalera. Per a això, cerca en un nombre de camins predeterminats. Si no els troba automàticament, se subratllarà en vermell el nom d'un fitxer de capçalera; en aquest cas, feu clic dret sobre seu per a indicar explícitament al KDevelop on trobar aquests fitxers i la informació que proporcionen.

NOTA
La configuració de la compleció automàtica es discuteix en aquesta secció d'aquest manual.

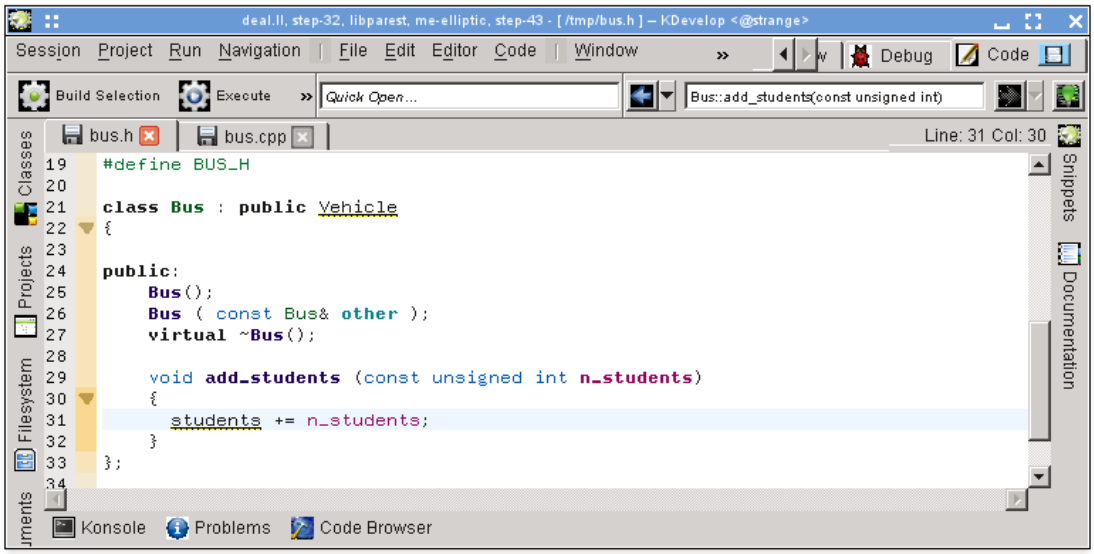
3.4.2 Afegir classes noves i implementar funcions de membre

El KDevelop inclou un assistent per a afegir classes noves. El procediment es descriu en Crear una classe nova. Una classe de C++ simple es pot crear escollint la plantilla de C++ bàsic des de la categoria `Classe`. En l'assistent, podem triar algunes funcions de membre predefinides, p. ex., un constructor buit, un constructor de còpia i un destructor.

Després de completar l'assistent, els fitxers nous seran creats i s'obriran a l'editor. El fitxer de capçalera ja conté guàrdies «include» i la classe nova conté totes les funcions de membre que hem seleccionat. Els dos passos següents serien documentar la classe, les seves funcions de membre i implementar-les. Més endavant parlarem sobre documentar les classes i les funcions. Per a implementar les funcions especials que ja hem afegit, simplement aneu a la pestanya **bus.cpp** on ja es proporciona l'esquelet de les funcions:

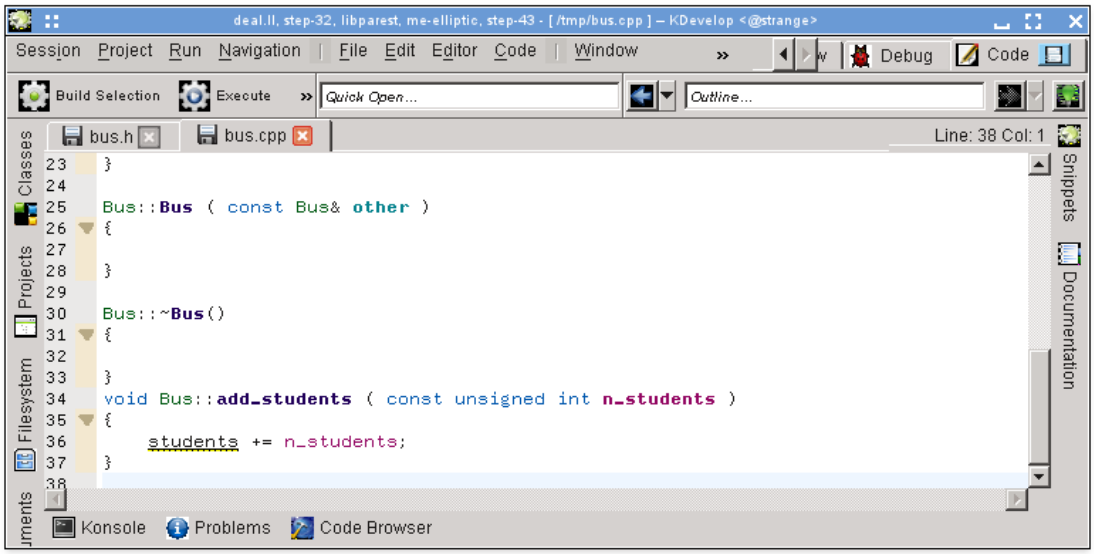


Per a afegir funcions noves de membres, torneu a la pestanya **bus.h** i afegiu-hi el nom d'una funció. Per exemple, afegirem això:



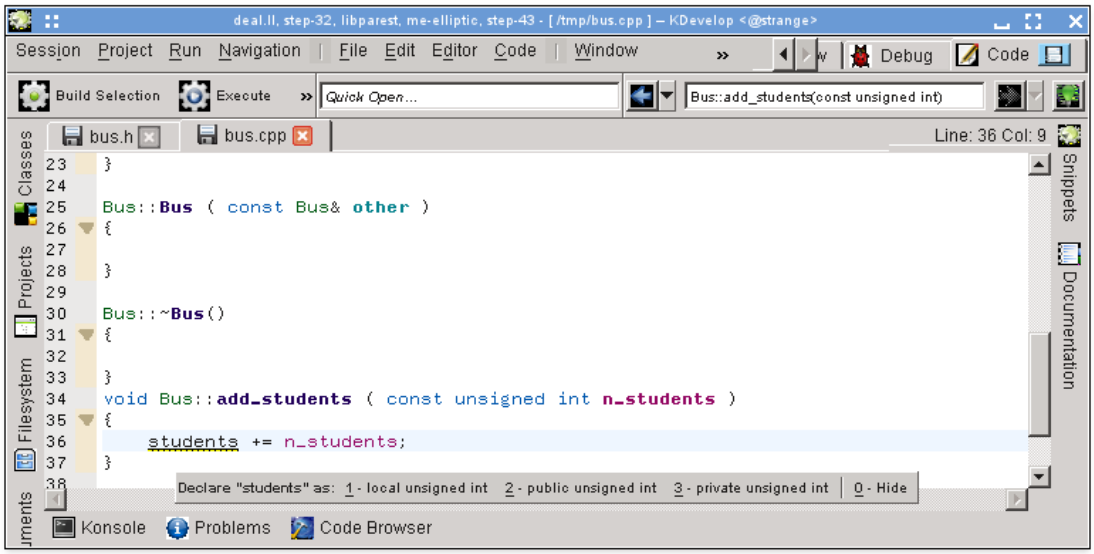

```
19  #define BUS_H
20
21  class Bus : public Vehicle
22  {
23
24  public:
25      Bus();
26      Bus ( const Bus& other );
27      virtual ~Bus();
28
29      void add_students (const unsigned int n_students)
30      {
31        students += n_students;
32      }
33  };
34
```

Tingueu en compte com ja he començat amb la implementació. No obstant això, en molts estils de codificació, la funció no s'ha d'implementar al fitxer de capçalera, sinó més aviat al fitxer .cpp corresponent. Per a això, situeu el cursor sobre el nom de la funció i seleccioneu **Codi** → **Moure a l'origen** o premeu **Ctrl-Alt-S**. Això eliminarà el codi entre claus del fitxer de capçalera (i el substituirà per un punt i coma si és necessari per a posar fi a la declaració de la funció) i es mourà al fitxer d'origen:

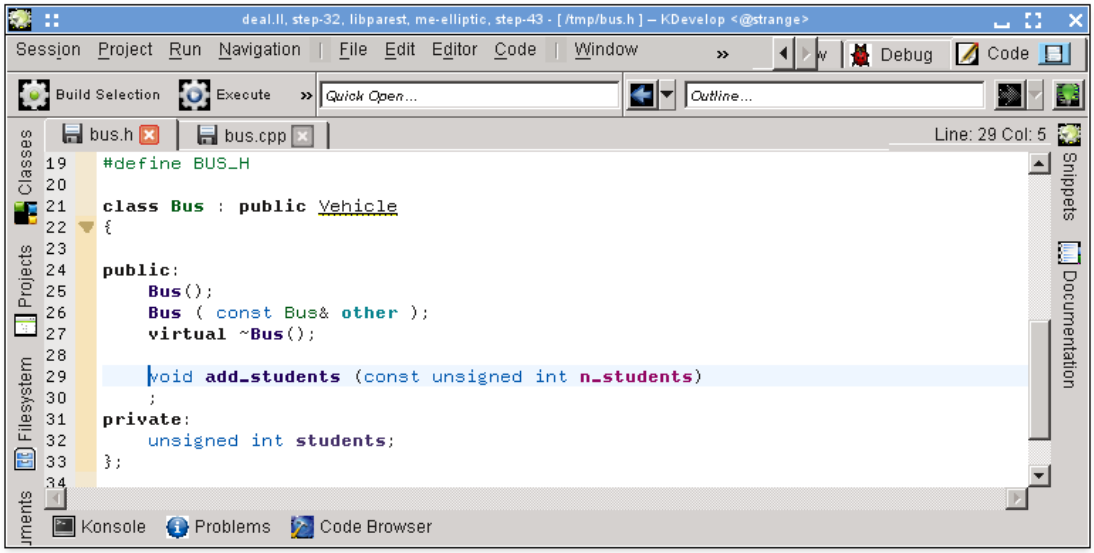



```
23  }
24
25  Bus::Bus ( const Bus& other )
26  {
27
28  }
29
30  Bus::~Bus()
31  {
32
33  }
34  void Bus::add_students ( const unsigned int n_students )
35  {
36      students += n_students;
37  }
38
```

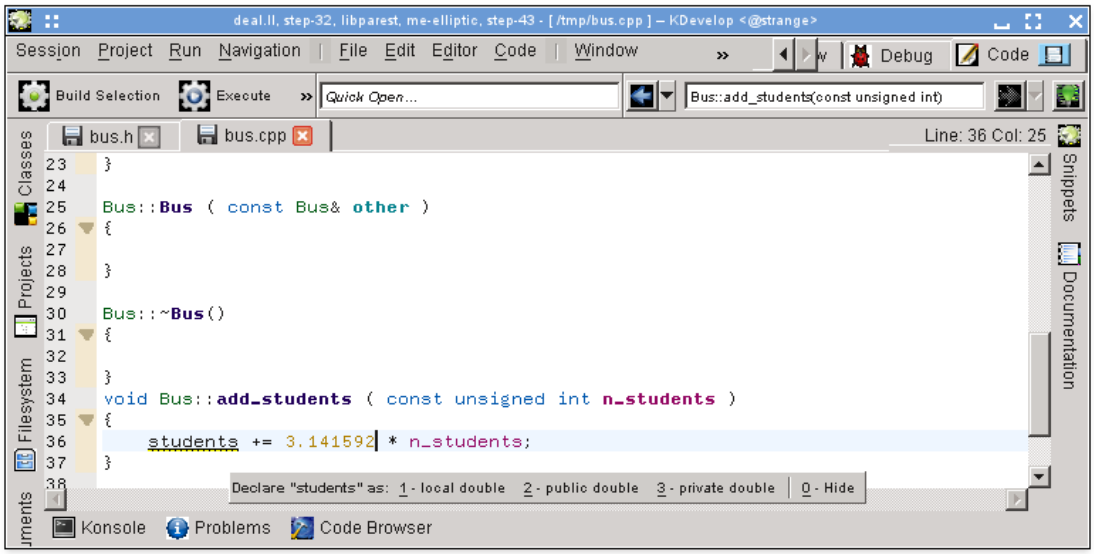
Tingueu en compte com només he començat a escriure i que vull implementar que la variable `students` probablement serà una variable de membre de la classe `Bus`, però que encara no l'he afegit. Tingueu en compte també com el KDevelop la subratlla per a deixar clar que no sap res sobre la variable. Però aquest problema es pot resoldre: Fent clic al nom de la variable s'obté el següent consell:



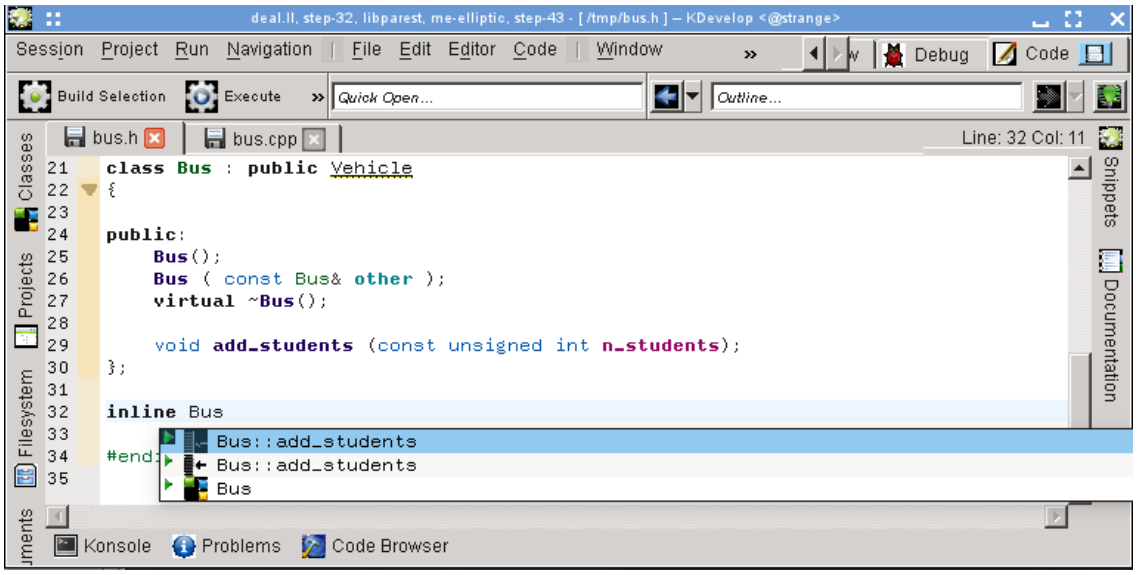
(El mateix es pot aconseguir fent clic dret sobre seu i seleccionant **Resoldre: Declara com a**). Permetem seleccionar «3 - unsigned int privat» (sigui amb el ratolí o prement **Alt-3**) i veiem com es mostra al fitxer de capçalera:



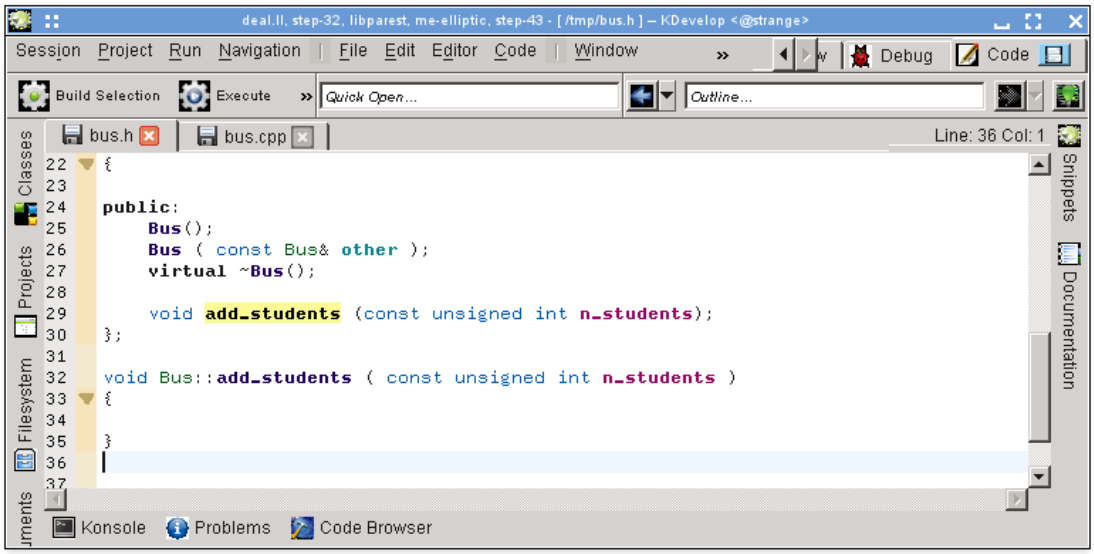
Val la pena assenyalar que el KDevelop extreu el tipus de la variable a declarar de l'expressió emprada per a inicialitzar-la. Per exemple, si haguéssim escrit l'addició de la manera següent bastant dubtosa, hauria suggerit declarar la variable com a tipus `double`:



Com a punt final: El mètode que utilitza **Codi** → **Mou a l'origen** no sempre insereix la funció nova de membre on probablement voleu. Per exemple, és possible que vulgueu que sigui marcada com a `inline` i posar-la a la part inferior del fitxer de capçalera. En un cas com aquest, escriviu la declaració i comenceu a escriure la definició de la funció de la manera següent:



El KDevelop oferirà automàticament totes les possibles complecions del que podria anar aquí. En seleccionar una de les dues entrades `add_students` s'obtindrà el següent codi que ja omple la llista d'arguments:




```
{

public:
    Bus();
    Bus ( const Bus& other );
    virtual ~Bus();

    void add_students (const unsigned int n_students);
};

void Bus::add_students ( const unsigned int n_students )
{

}
```

NOTA

En l'exemple, l'acceptació d'una de les opcions de l'eina de compleció automàtica ofereix la signatura correcta, però malauradament suprimeix el marcador `inline` que ja s'ha escrit. Això ha estat informat com a error 274245 del KDevelop.

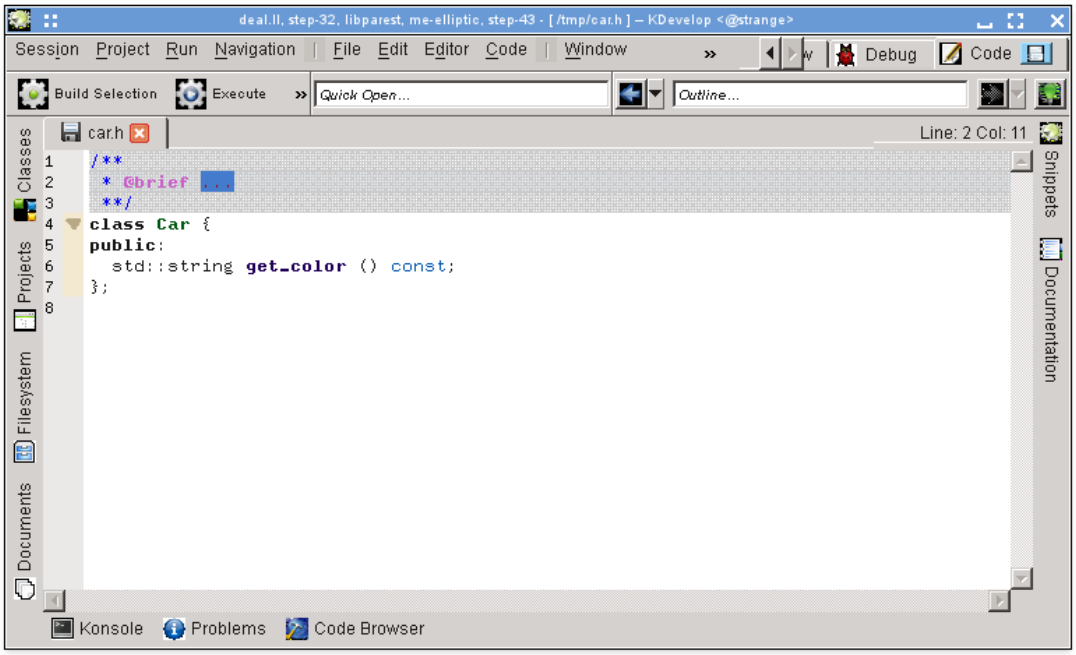
3.4.3 Documentar les declaracions

El bon codi està ben documentat, tant en l'àmbit de la implementació dels algoritmes dins de les funcions com en l'àmbit de la interfície -és a dir, les classes, les funcions (de membre i globals) i les variables (de membre i globals)- s'han de documentar per a explicar les seves intencions, possibles valors dels arguments, condicions prèvies i posteriors, etc. Pel que fa a la documentació de la interfície, el doxygen s'ha convertit en l'estàndard de facto per a donar format als comentaris que després es podran extreure i visualitzar en pàgines de cerca.

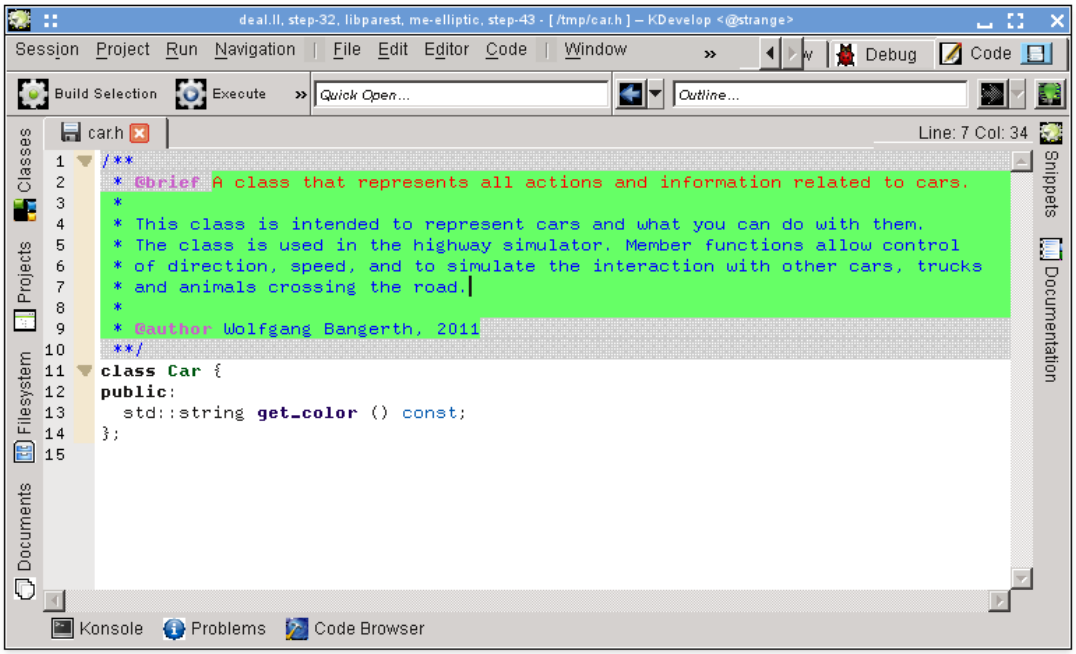
El KDevelop implementa aquest estil de comentaris proporcionant una drecera per a generar el marc de treball dels comentaris que documenten una funció de classe o funció de membre. Per exemple, suposeu que ja heu escrit el codi:

```
class Car {
  public:
    std::string get_color () const;
};
```

Ara voleu afegir documentació tant a la classe i a la funció de membre. Per a això, situeu el cursor a la primera línia i seleccioneu **Codi** → **Documenta la declaració** o premeu **Alt-Maj-D**. El KDevelop respondrà amb el següent:

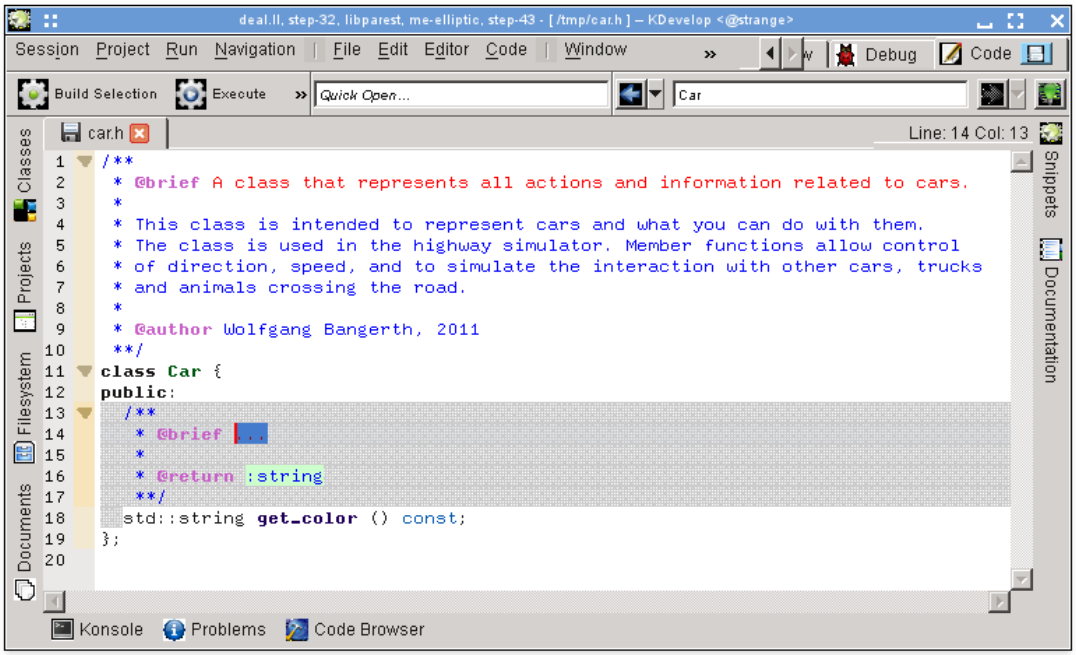


El cursor ja es troba a l'àrea en gris perquè ompliu la descripció curta (després de la paraula clau `@brief` de «doxygen») d'aquesta classe. A continuació, podeu continuar afegint documentació a aquest comentari, la qual donarà un resum més detallat del que fa la classe:

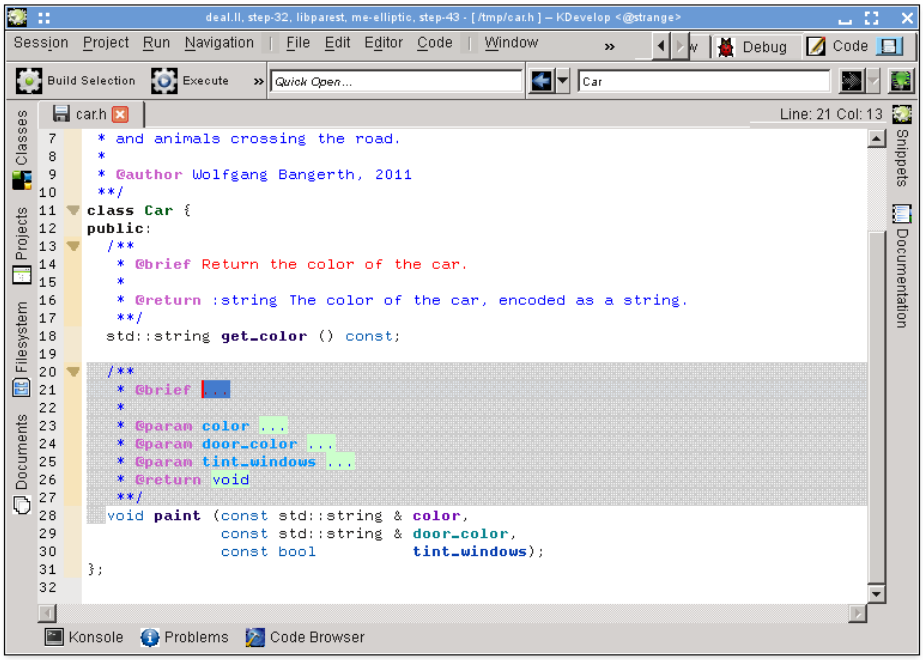


Mentre l'editor resti dins del comentari, el text del comentari serà ressaltat en verd (el ressaltat desapareixerà una vegada moveu el cursor fora del comentari). En arribar al final d'una línia, premeu **Retorn** i el KDevelop iniciarà automàticament una línia nova que començarà amb un asterisc i posarà al cursor un caràcter de sagnat.

Ara documentarem la funció de membre, un cop més, posant el cursor a la línia de la declaració i seleccionant **Codi** → **Documenta la declaració** o prement **Alt-Maj-D**:



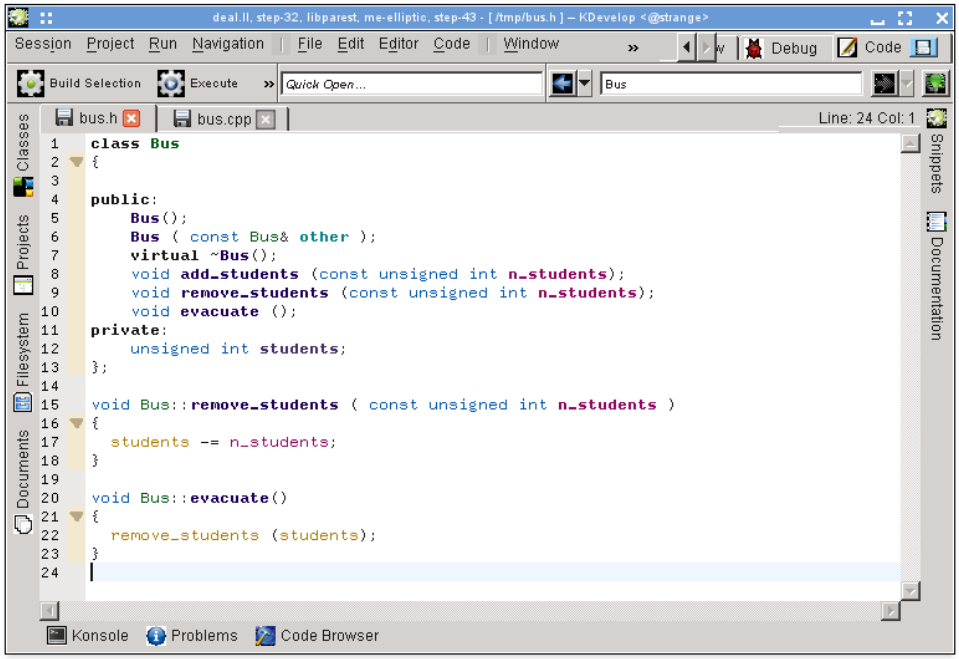
Una vegada més, el KDevelop generarà automàticament l'esquelet d'un comentari, inclosa la documentació per a la funció «itself», així com el seu tipus de retorn. En el cas actual, el nom de la funció és força autoexplicatiu, però moltes vegades els arguments de la funció poden no ser i han de ser documentats de manera individual. Per a il·lustrar-ho, considerem una funció una mica més interessant i el comentari que generarà automàticament el KDevelop:



Aquí, p. ex., el comentari suggerit ja conté tots els camps Doxygen per als paràmetres individuals.

3.4.4 Canviar el nom de les variables, funcions i classes

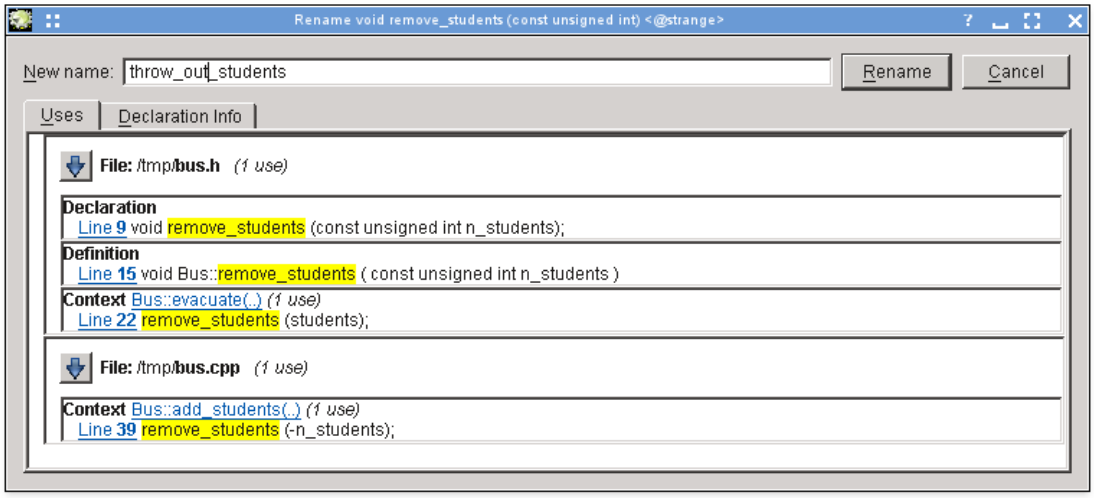
A vegades, un vol canviar el nom d'una funció, classe o variable. Per exemple, diguem que ja tenim això:



Llavors ens adonem que estem descontents amb el nom `remove_students` i seria millor anomenar-la `throw_out_students`. Podríem fer una cerca de substitució pel nom, però això té dos inconvenients:

- La funció es pot utilitzar en més d'un fitxer.
- Realment només volem canviar el nom d'aquesta funció i no tocar les funcions que poden tenir el mateix nom però es declaren en altres classes o espais de nom.

Ambdós problemes es poden resoldre movent el cursor sobre qualsevol de les ocurrències del nom de la funció i seleccionant **Codi** → **Reanomena la declaració** (o fent clic dret sobre el nom i seleccionant **Canvia Bus::remove_students**). Això mostrarà un diàleg on podreu introduir el nom nou de la funció i on també podreu veure tots els llocs on s'utilitza en realitat la funció:

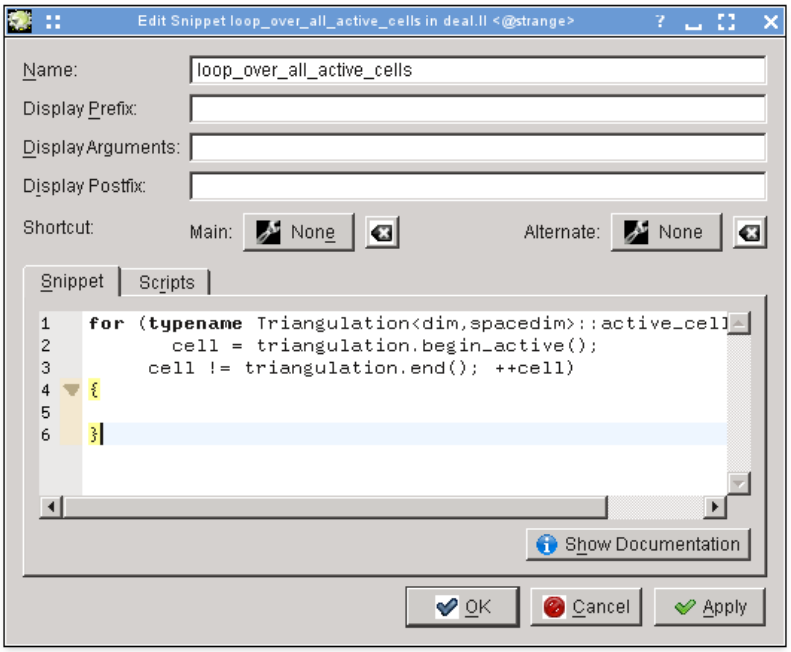


3.4.5 Retalls de codi

La majoria dels projectes tenen peces de codi que s'ha d'escriure sovint al codi font. Els exemples són: pels escriptors del compilador, un bucle sobre totes les instruccions; pels escriptors de la interfície d'usuari, comprovar que l'entrada de l'usuari és vàlida i si no ho és, obrir un quadre d'error; en el projecte de l'autor d'aquestes línies, això seria codi del tipus

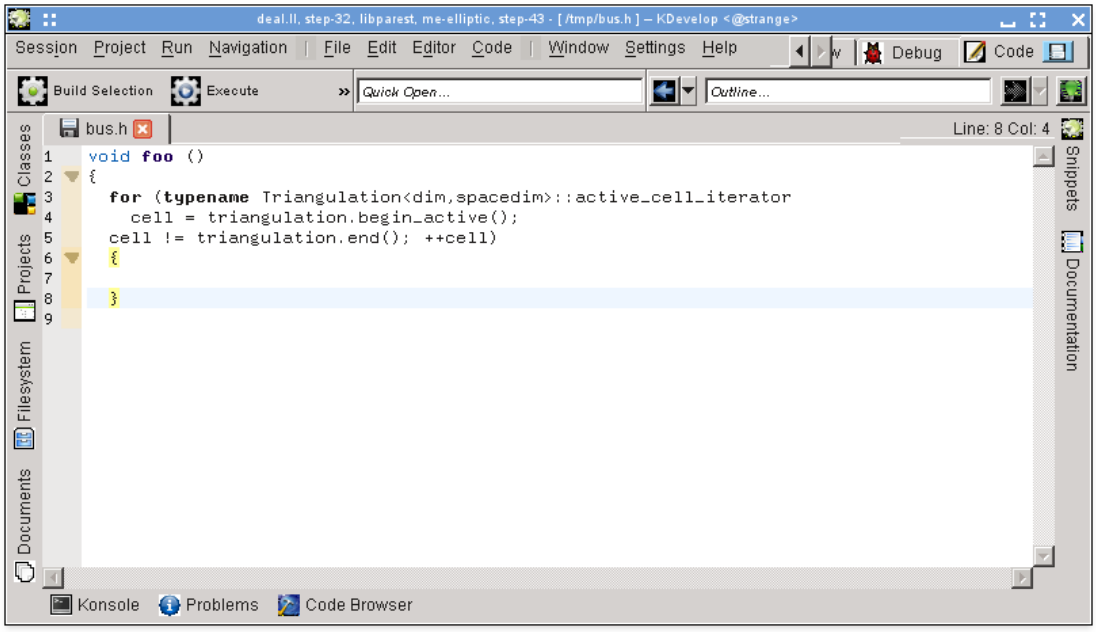
```
for (typename Triangulation::active_cell_iterator
       cell = triangulation.begin_active();
     cell != triangulation.end(); ++cell)
  ... do something with the cell ...
```

En lloc d'escriure aquest tipus de text una vegada darrere l'altra (amb tots els errors tipogràfics repetitius que s'introdueixin), l'eina **Retalls** del KDevelop us hi pot ajudar. Per a això, obriu la vista d'eina (vegeu Eines i vistes, si el botó corresponent no es troba ja en el perímetre de la vostra finestra). Després feu clic al botó «Afegeix un repositori» (un terme poc apropiat -el qual permet crear una col·lecció amb nom de retalls de codi font d'un tipus en particular, p. ex., codi font en C++-) i crear un repositori buit. A continuació, feu clic a la icona per a afegir un retall de codi, per a obtenir un diàleg com el següent:



> **NOTA**
> El nom d'un retall no pot contenir espais ni altres caràcters especials, ja que s'ha d'assemblar a una funció normal o nom de variable (per raons que s'aclariran en el següent paràgraf).

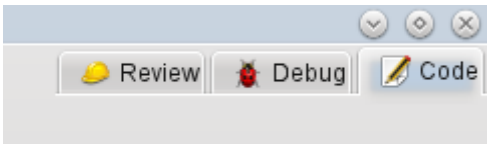
Per a emprar un retall ja definit, quan esteu editant codi, simplement escriviu el nom del retall com ho faríeu amb qualsevol altra nom de funció o de variable. Aquest nom restarà disponible per a compleció automàtica -el que significa que no hi ha cap problema en l'ús d'un nom llarg i descriptiu per a un fragment de codi com l'anterior- i quan accepteu el suggeriment per a la compleció automàtica (p. ex., només prement la tecla **Retorn**), la part ja introduïda del nom del retall serà substituïda per l'expansió completa del retall i serà sagnat adequadament:



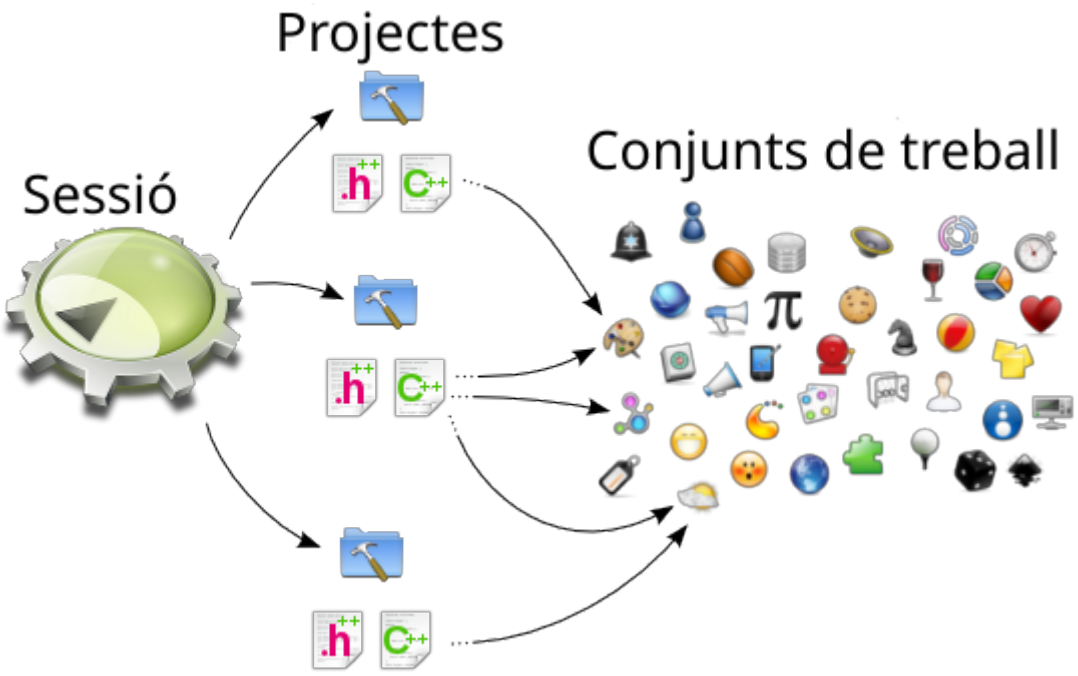
Tingueu en compte que perquè això funcioni, la vista d'eina **Retalls** no s'haurà d'obrir o estar visible: només es necessita sempre la vista d'eina per a definir retalls nous. Una manera alternativa, encara que menys còmoda, per a expandir un retall és simplement fer clic en la vista d'eina respectiva.

> NOTA
> Els retalls són molt més poderosos del que acabem d'explicar. Per a una descripció completa del que podeu fer amb ells, vegeu la documentació detallada de l'eina Retalls.

3.5 Modes i conjunts de treball

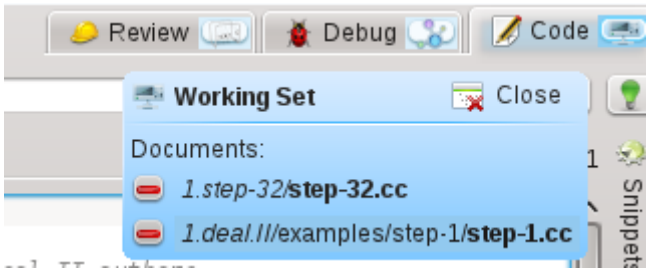


Si heu arribat fins aquí, doneu una ullada a la part superior dreta de la finestra principal del KDevelop: Com es mostra a la imatge, veureu que hi ha tres **modes** en els quals pot estar el KDevelop: **Codi** (el mode es discuteix en el capítol actual en treballar amb el codi font), **Depuració** (vegeu Depurar programes) i **Revisió** (vegeu Treballar amb sistemes de control de versions).

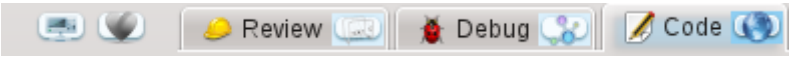


Cada mode té el seu propi conjunt d'eines que s'apilen en tot el perímetre, i cada mode també té un *conjunt de treball* dels fitxers i documents actualment oberts. A més, cadascun d'aquests conjunts de treball està associat amb una sessió actual, és a dir, tenim la relació mostrada anteriorment. Tingueu en compte que els fitxers en el conjunt de treball provenen de la mateixa sessió, però poden provenir de diferents projectes que formen part de la mateixa sessió.

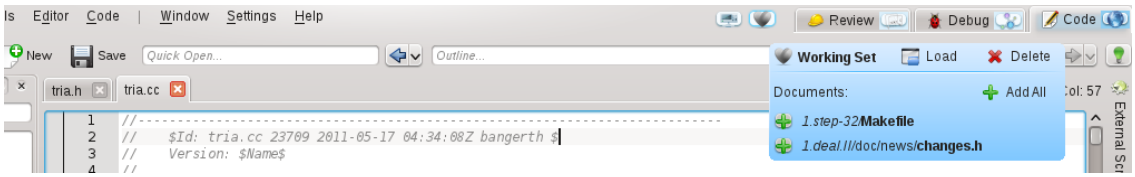
Si obriu el KDevelop per primera vegada, el conjunt de treball estarà buit -no hi haurà cap fitxer obert-. Però a mesura que s'obrin els fitxers per a l'edició (depuració o revisió en els altres modes) el conjunt de treball creixerà. El fet que el vostre conjunt de treball no estigui buit s'indica mitjançant un símbol a la pestanya, com es mostra a continuació. Us adonareu que cada vegada que es tanca el KDevelop i després comença de nou, el conjunt de treball es desa i es restaura, és a dir, obteniu el mateix conjunt de fitxers oberts.



Si passeu el ratolí sobre el símbol per al conjunt de treball, obtindreu un consell que us mostrarà quins fitxers estan oberts en aquest conjunt de treball (en aquest cas: els fitxers `step-32.cc` i `step-1.cc`). En fer clic al signe menys en vermell es tancarà la pestanya per al fitxer corresponent. Potser el més important, en fer clic al botó anomenat corresponent us permet tancar tot el treball creat alhora (és a dir, per a tancar tots els fitxers oberts). La importància de tancar un conjunt de treball, però, resideix en el fet que no es limita a tancar tots els fitxers, sinó que realment desa el conjunt de treball i n'obre un de nou buit. Això es pot veure a continuació:



Noteu els dos símbols a l'esquerra de les tres pestanyes de mode (el cor i el símbol no identificable a la seva esquerra). Cadascun d'aquests dos símbols representa un conjunt de treball desat, a més del conjunt de treball actualment obert. Si passeu el ratolí sobre el símbol del cor, obtindreu quelcom com això:



Us mostra que el conjunt de treball corresponent conté dos fitxers i els seus noms de projecte corresponent: `Makefile` i `changes.h`. En fer clic a **Carrega** es tancarà i desarà el conjunt de treball actual (com es mostra aquí, té oberts els fitxers `tria.h` i `tria.cc`) i s'obrirà en el seu lloc el conjunt de treball seleccionat. També podeu suprimir de forma permanent un conjunt de treball, el qual suprimirà el conjunt de conjunts de treball desats.

3.6 Algunes dreceres de teclat útils

L'editor del KDevelop segueix les dreceres de teclat estàndard per a totes les operacions d'edició habituals. No obstant això, també admet amb una sèrie d'operacions més avançades en editar codi font, algunes de les quals estan vinculades a determinades combinacions de tecles. Les següents sovint són especialment útils:

Saltar a través del codi	
Ctrl-Alt-O	Obertura ràpida del fitxer: introduir part d'un nom de fitxer i seleccionar tots els fitxers als arbres del directori dels projectes a la sessió actual que coincideixin amb la cadena. A continuació, s'obrirà el fitxer.
Ctrl-Alt-C	Obertura ràpida d'una classe: introduir part d'un nom de classe i seleccionar entre tots els noms de les classes que coincideixen. El cursor saltarà a la declaració de la classe.
Ctrl-Alt-M	Obertura ràpida d'una funció: introduir part d'un nom de funció (membre) i seleccionar entre tots els noms que coincideixin. Tingueu en compte que la llista mostrarà tant les declaracions com les definicions i el cursor després saltarà a l'element seleccionat.
Ctrl-Alt-Q	Obertura ràpida universal: escriure quelcom (nom de fitxer, nom de la classe, nom de la funció) i obtindreu una llista de tot el que coincideixi per a triar.

Ctrl-Alt-N	Esquema: proporciona una llista de totes les coses que estan succeint en aquest fitxer, p. ex., les declaracions de classe i la definició de les funcions.
Ctrl-,	Va a la definició d'una funció si el cursor es troba actualment en una declaració de funció.
Ctrl-.	Va a la declaració d'una funció o variable si el cursor es troba actualment en una definició de funció.
Ctrl-Alt-Av Pàg	Salta a la funció següent.
Ctrl-Alt-Re Pàg	Salta a la funció anterior.
Ctrl-G	Va a la línia.

Cercar i substituir	
Ctrl-F	Cerca.
F3	Cerca la següent.
Ctrl-R	Substitueix.
Ctrl-Alt-F	Cerca i substitueix en múltiples fitxers.

Altres coses	
Ctrl-_	Contrau un nivell: suprimir aquest bloc de la vista, p. ex., si voleu centrar-vos en una visió més gran dins d'una funció.
Ctrl-+	Expandeix un nivell: desfà l'acció de contraure.
Ctrl-D	Descomenta el text seleccionat o la línia actual.
Ctrl-Maj-D	Comenta el text seleccionat o la línia actual.
Alt-Maj-D	Documenta la funció actual. Si el cursor està en una declaració de funció o classe, després de prémer aquesta tecla es crearà un comentari a l'estil doxygen, contenint una llista de tots els paràmetres, valors de retorn, etc.
Ctrl-T	Intercanvia el caràcter actual i l'anterior.
Ctrl-K	Suprimeix la línia actual (nota: això no és com a «emacs» -suprimeix des d'aquí fins al final de la línia-).

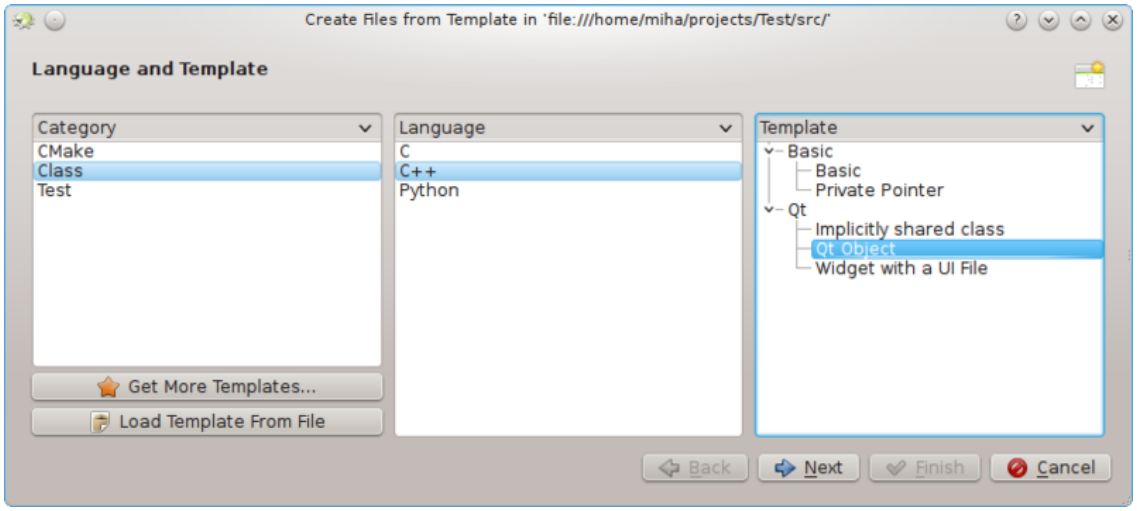
Capítol 4

Generar el codi amb plantilles

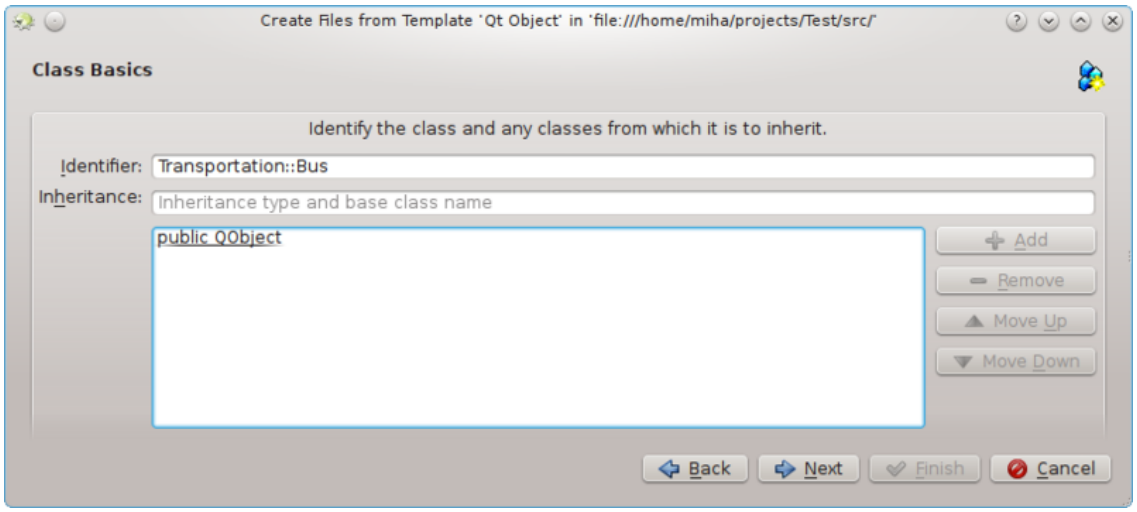
El KDevelop utilitza plantilles per a la generació dels fitxers de codi font i per a evitar escriure codi repetitiu.

4.1 Crear una classe nova

L'ús més comú per a la generació de codi probablement és escrivint classes noves. Per a crear una classe nova en un projecte existent, feu clic dret en una carpeta de projecte i escolliu **Crea des d'una plantilla...**. El mateix diàleg es pot iniciar des del menú fent clic a **Fitxer** → **Nou des d'una plantilla...**, però emprar una carpeta de projecte té l'avantatge d'establir un URL base pels fitxers de sortida. Escolliu `Classe` a la vista per a la selecció de la categoria, el llenguatge desitjat i la plantilla en les altres dues vistes. Després de seleccionar una plantilla per a la classe, haureu d'especificar els detalls de la classe nova.

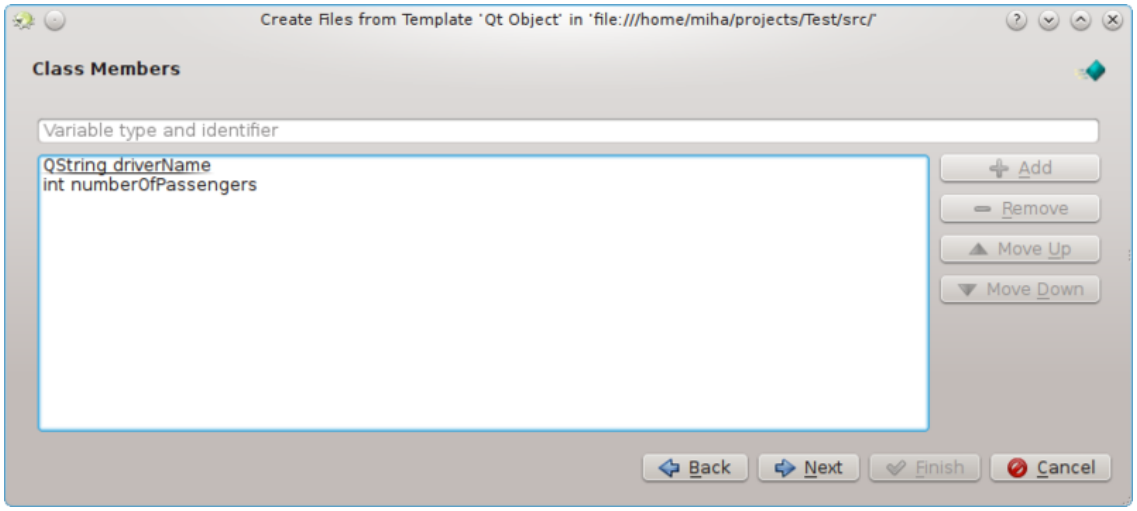


Primer heu d'especificar un identificador per a la classe nova. Això pot ser un nom únic (com `Bus`) o un identificador complet amb espais de nom (com `Transportation::Bus`). En aquest últim cas, el KDevelop analitzarà l'identificador i separarà correctament els espais de nom del nom real. A la mateixa pàgina, podeu afegir classes base per a la classe nova. Possiblement us adonareu que algunes plantilles trien una classe base pel seu compte, sou lliure d'eliminar i/o afegir altres bases. Aquí heu d'escriure la declaració d'herència completa, la qual és dependent del llenguatge, com `public QObject` per a C++, `extends AlgunaClasse` per a PHP o simplement el nom de la classe per a Python.



A la pàgina següent, se us ofereix una selecció dels mètodes virtuals des de totes les classes heretades, així com alguns constructors, destructors i operadors per omissió. En marcar la casella de selecció situada al costat d'una signatura de mètode s'implementarà aquest mètode en la classe nova.

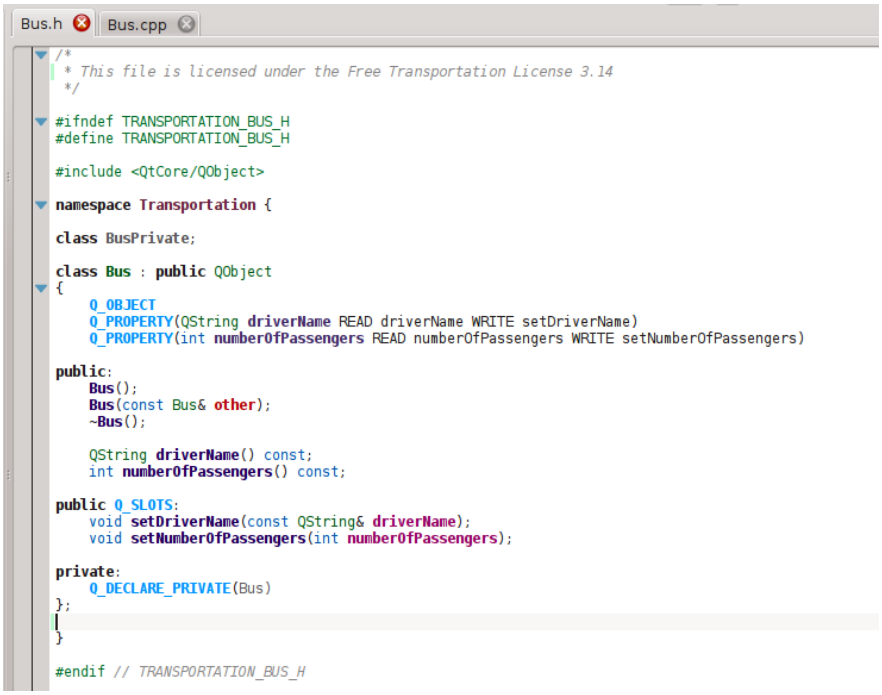
En fer clic a **Següent** apareixerà una pàgina on podreu afegir membres a una classe. Depenent de la plantilla seleccionada, aquests podran aparèixer en la classe nova com a variables de membre, o la plantilla podrà crear propietats pels seus respectius «setters» i «getters». En un llenguatge on s'han de declarar els tipus de les variables, com ara C++, haureu d'especificar el tipus i el nom del membre, com ara `int número` o `QString nom`. En altres llenguatges, és possible deixar de banda el tipus, però és una bona pràctica introduir-lo sempre, atès que la plantilla seleccionada encara podria fer-ne algun ús.



En les següents pàgines, podeu triar una llicència per a la classe nova, establiu qualsevol de les opcions personalitzades que necessiteu per a la plantilla seleccionada, i configureu les ubicacions de sortida per a tots els fitxers generats. En fer clic a **Finalitza**, completareu l'assistent i es crearà la classe nova. Els fitxers generats seran oberts a l'editor, així que podreu començar a afegir codi de seguida.

Després de crear una classe nova de C++, se us donarà l'opció d'afegir la classe a l'objectiu d'un projecte. Trieu l'objectiu des de la pàgina del diàleg, o descarteu la pàgina i afegiu els fitxers a un objectiu de forma manual.

Si heu triat la plantilla `Qt Object`, marcant alguns dels mètodes per omissió i afegint dues variables de membre, la sortida s'hauria d'assemblar a la següent imatge.



Podeu veure que els membres de dades es converteixen en propietats de Qt, amb funcions d'accés i les macros Q_PROPERTY. D'altra banda, els arguments per a les funcions «setter» només són passats com a referències a les constants quan és apropiat. A més, es declara una classe privada i un punter privat amb Q_DECLARE_PRIVATE. Tot això és realitzat per la plantilla, la tria d'una plantilla diferent en el primer pas podria canviar completament la sortida.

4.2 Crear una prova unitària nova

Encara que la majoria dels marcs de treball per a proves requereixen que cada prova sigui també una classe, el KDevelop inclou un mètode per a simplificar la creació de les proves unitàries. Per a crear una prova nova, feu clic dret en una carpeta de projecte i trieu **Crea des d'una plantilla...**. A la pàgina per a la selecció de la plantilla, trieu `Prova` com a categoria, després trieu el vostre llenguatge de programació i la plantilla, i feu clic a **Següent**.

Se us demanarà el nom de la prova i una llista de casos de prova. Pels casos de prova, n'hi ha prou amb especificar una llista de noms. Alguns marcs de treball per a proves unitàries, com PyUnit i PHPUnit, exigeixen que els casos de prova comencin amb un prefix especial. Al KDevelop, la plantilla és la responsable d'afegir el prefix, de manera que aquí no haureu de prefixar els casos de prova. Després feu clic a **Següent**, especifiqueu les ubicacions de la llicència i de la sortida pels fitxers generats, i la prova serà creada.

Les proves unitàries creades d'aquesta manera no es poden afegir a qualsevol objectiu de forma automàtica. Si empreu CTest o algun altre marc de treball per a les proves, assegureu-vos d'afegir els fitxers nous a un objectiu.

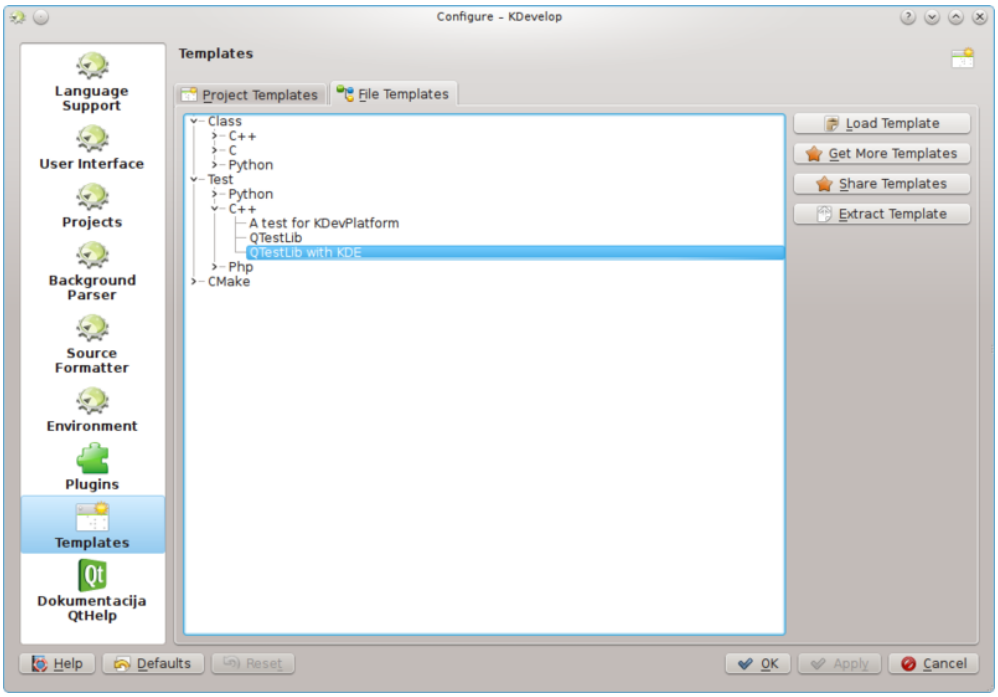
4.3 Altres fitxers

Mentre que les classes i les proves unitàries reben una especial atenció a la generació del codi a partir de plantilles, el mateix mètode es pot ser emprar per a qualsevol tipus de fitxers de codi font. Per exemple, es podria emprar una plantilla per a un mòdul Find de CMake o un fitxer «.desktop». Això es pot fer seleccionant **Crea des d'una plantilla...** i seleccionar la categoria i

plantilla desitjades. Si la categoria seleccionada no és ni `Classe` ni `Prova`, només tindreu l'opció de triar la llicència, les opcions personalitzades especificades per la plantilla i les ubicacions pels fitxers de sortida. Igual que amb les classes i les proves, el fet de finalitzar l'assistent generarà els fitxers i els obrirà a l'editor.

4.4 Gestionar les plantilles

Des de l'assistent **Fitxer** → **Nou des d'una plantilla...**, també podreu baixar plantilles de fitxers addicionals fent clic al botó **Obtén més plantilles...**. Aquest obrirà un diàleg «Obtén les novetats candents», des d'on podreu instal·lar plantilles addicionals, així com actualitzar-les o eliminar-les. També hi ha un mòdul de configuració per a les plantilles, al qual es pot arribar fent clic a **Arranjament** → **Configura el KDevelop...** → **Plantilles**. A partir d'aquí, podreu gestionar tant les plantilles de fitxers (s'ha explicat anteriorment) com les plantilles de projectes (emprades per a la creació de projectes nous).



Per descomptat, si cap de les plantilles disponibles s'adapta al vostre projecte, sempre podeu crear-ne de noves. La manera més senzilla probablement és copiant i modificant una plantilla existent, tot i que disposeu d'una breu guia d'aprenentatge i un llarg document d'especificació per a ajudar-vos. Per a copiar una plantilla instal·lada, obriu el gestor de plantilles fent clic a **Arranjament** → **Configura el KDevelop...** → **Plantilles**, seleccioneu la plantilla que voleu copiar i després feu clic al botó **Extreu la plantilla**. Seleccioneu una carpeta de destinació, feu clic a **D'acord**, i el contingut de la plantilla s'extraurà a la carpeta seleccionada. Ara podeu editar la plantilla obrint els fitxers extrets i modificant-los. Una vegada acabeu, podreu importar la plantilla nova al KDevelop obrint el gestor de plantilles, activant la pestanya corresponent (siguin **Plantilles de projectes** o **Plantilles de fitxers**) i feu clic a **Carrega una plantilla**. Obriu el fitxer de descripció de la plantilla, el qual és el que té el sufix «.kdevtemplate» o «.desktop». El KDevelop comprimirà els fitxers dins d'un fitxer de plantilles i l'importarà.

> **NOTA**
> En copiar una plantilla existent, assegureu-vos de canviar el seu nom abans d'importar-la una altra vegada. En cas contrari, sobreescriureu el model antic o acabareu amb dues plantilles amb noms idèntics. Per a canviar el nom d'una plantilla, canvieu el nom del fitxer de descripció a quelcom únic (però mantenint el sufix) i canvieu l'entrada `Nom` al fitxer de descripció.

Si voleu escriure una plantilla des de zero, podeu començar amb una plantilla de classe per a C++ de mostra creant un projecte nou i seleccionant `Plantilla de classe en C++` a la categoria `KDevelop`.

Capítol 5

Construir (compilar) projectes amb els «Makefile» personalitzats

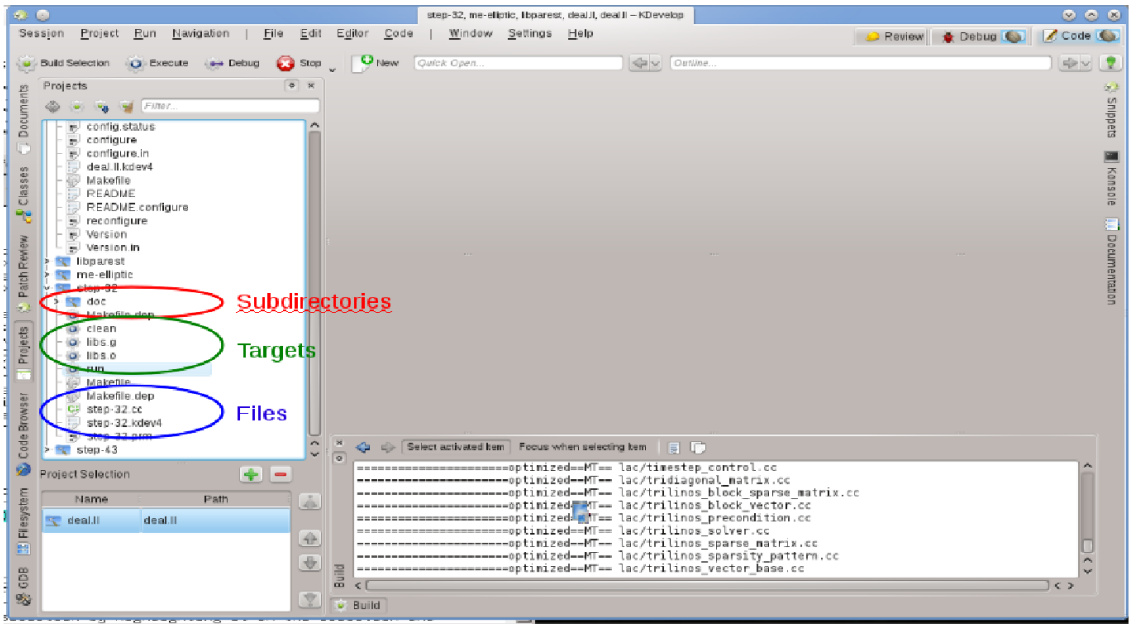
Molts projectes descriuen com s'han de compilar els fitxers d'origen (i quins fitxers s'han de tornar a compilar un cop s'ha fet algun canvi a la font o un fitxer de capçalera) emprant els fitxers «Makefile», els quals són interpretats pel programa **make** (vegeu, p. ex., el make del GNU). Per a projectes senzills, sovint és molt més fàcil configurar manualment un fitxer d'aquest tipus. Els projectes més grans, sovint integren els seus «Makefile» amb les **autotools de GNU** (autoconf, autoheader, automake). En aquesta secció, simplement assumirem que teniu un Makefile per al vostre projecte i volem ensenyar al KDevelop com interactuar amb ell.

> NOTA
> El KDevelop 4.x no sap res de les **autotools de GNU** en el moment d'escriure aquesta secció. Si el vostre projecte les utilitza, haureu d'executar `./configure` manualment o qualsevol de les altres ordres relacionades des d'una línia d'ordres. Si voleu fer-ho dins del KDevelop, obriu l'eina **Konsole** (si cal afegir-la al perímetre de la finestra principal, empreu el menú **Finestres** → **Afegeix una vista d'eina**), la qual us oferirà un intèrpret d'ordres i executeu `./configure`.

El primer pas és ensenyar al KDevelop els objectius als vostres «Makefile». Hi ha dues maneres de fer-ho: seleccionant objectius «Makefile» individualment, o triant un conjunt d'objectius que potser voleu construir amb freqüència. Per a tots dos enfocaments, obriu l'eina **Projectes** fent clic al botó **Projectes** en el perímetre de la finestra principal del KDevelop (si no disposeu d'aquest botó, vegeu anteriorment com afegir el botó d'una eina). La finestra de l'eina **Projectes** té dues parts: La meitat superior - anomenada **Projectes** - enumera tots els vostres projectes i permet ampliar els arbres de directoris subjacents. La meitat inferior - anomenada **Selecció de projectes** - enumera un subconjunt d'aquests projectes que es construiran si trieu l'element de menú **Projecte** → **Construeix la selecció** o premeu la tecla **F8**. Més endavant tornarem a aquesta qüestió.

5.1 Construir cadascun dels objectius «Makefile»

A la part superior de la vista del projecte, s'expandeixi el subarbre per a un projecte, diguem que aquell per al qual voleu executar un objectiu «Makefile» en particular. Això li donarà icones per a: (i) els directoris sota aquest projecte, (ii) els fitxers en el directori de nivell superior per a aquest projecte, (iii) els objectius «Makefile» que pot identificar el KDevelop. Aquestes categories es mostren a la imatge de la dreta. Recordeu que el KDevelop *entén* la sintaxi dels «Makefile» fins a cert punt, així que us pot oferir objectius definits en aquest «Makefile» (encara que aquesta entesa tingui els seus límits si els objectius són compostos o implícits).



Per a construir qualsevol dels objectius que hi figuren, feu clic amb el botó dret del ratolí i seleccioneu **Construeix**. Per exemple, fer-ho amb l'objectiu de «neteja» simplement executarà «make clean». Podreu veure què està succeint a la subfinestra que apareixerà anomenada **Construcció**, mostrant l'ordre i la sortida. (Aquesta finestra es correspon amb l'eina **Construeix**, de manera que la podeu tancar i després tornar a obrir emprant el botó de l'eina **Construeix** sobre el perímetre de la finestra principal. Es mostra a la part inferior dreta de la imatge).

5.2 Seleccionar un conjunt d'objectius «Makefile» per a repetir la construcció

El fet de fer clic dret sobre objectius «Makefile» individuals cada vegada que vulgueu construir alguna cosa us farà perdre un temps preciós. En comptes d'això, ens agradaria tenir objectius individuals per a un o més dels projectes en la sessió que podem construir en diverses ocasions sense gaire feina del ratolí. Aquí és on entra en joc el concepte de «Construir els objectius seleccionats»: es tracta d'una col·lecció d'objectius «Makefile» que es construeixen l'un darrere l'altre cada vegada que es prem el botó **Construeix la selecció** a la llista de botons en la part superior, seleccioneu l'element de menú **Projecte** → **Construeix la selecció**, o premeu la tecla de funció **F8**.

La llista dels objectius «Makefile» seleccionats es mostra a la meitat inferior de la vista d'eina **Projectes**.

Per omissió, la selecció conté tots els projectes, però la podeu canviar. Per exemple, si la vostra llista de projectes conté tres projectes (una biblioteca base L i dues aplicacions A i B), però actualment només treballeu en el projecte A, llavors possiblement voldreu suprimir el projecte B de la selecció, ressaltant-lo en la selecció i prement el botó . A més, probablement voldreu assegurar-vos que la biblioteca L serà construïda abans que el projecte A movent les entrades a la selecció cap amunt i cap avall amb els botons a la dreta de la llista. També podeu obtenir un objectiu «Makefile» determinat en la selecció fent clic dret sobre seu i seleccionant **Afegeix al conjunt de construcció**, o simplement seleccionant-lo i prement el botó just a sobre de la llista d'objectius seleccionats.

El KDevelop permet configurar què fer cada vegada que es construeix la selecció. Per això, empreu l'element de menú **Projecte** → **Obre la configuració**. És on podreu, p. ex., seleccionar el

nombre de tasques simultànies que s'hauran d'executar per a «make» -si l'ordinador té, diguem-ne, 8 nuclis de processador, llavors introduir 8 en aquest camp seria una bona opció-. En aquest diàleg, l'**Objectiu de «make» per omissió** és un objectiu «Makefile» emprat per a *tots* els objectius en la selecció.

5.3 Què fer amb els missatges d'error

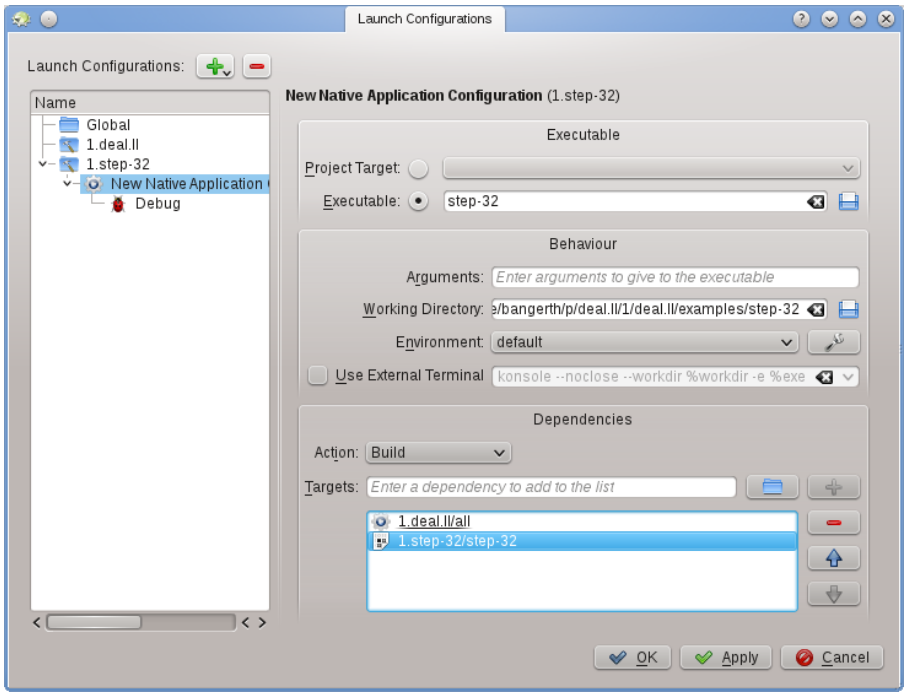
Si el compilador troba un missatge d'error, simplement feu clic a la línia amb el missatge d'error i l'editor saltarà a la línia (i si es disposa de la columna), on s'ha informat de l'error. Depenent del missatge d'error, el KDevelop també pot oferir diverses accions possibles per a corregir l'error, p. ex., la declaració d'una variable sense declarar si ha trobat un símbol desconegut.

Capítol 6

Executar programes al KDevelop

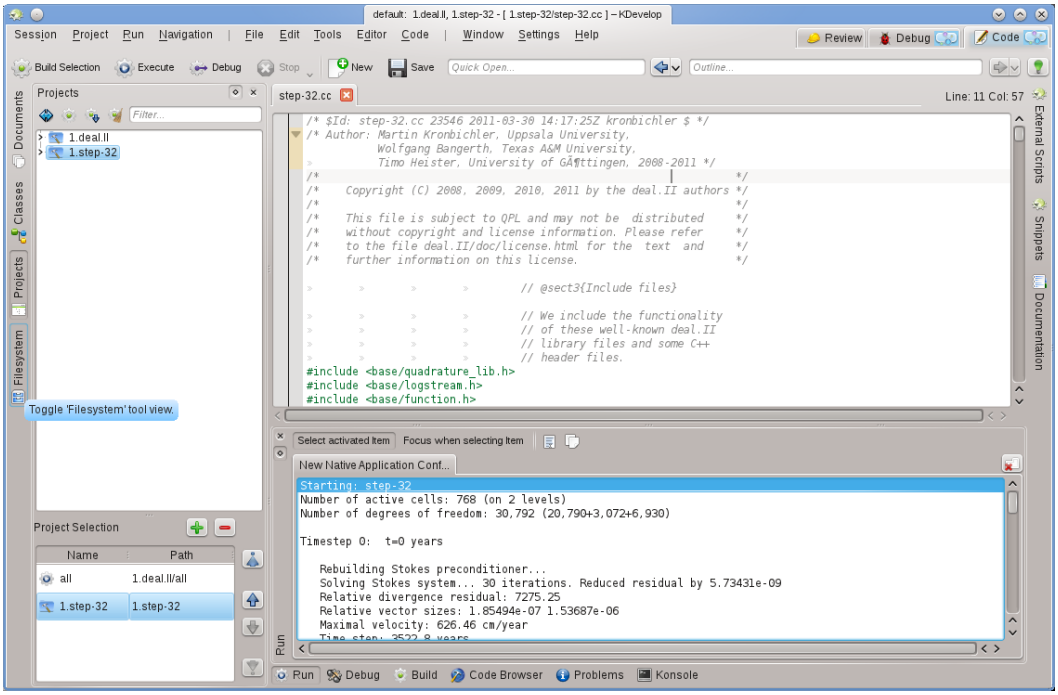
Una vegada que heu construït un programa, l'haureu d'executar. Per a fer-ho, haureu de configurar el *llançador* per als vostres projectes. Un *llançament* consisteix en el nom d'un executable, un conjunt de paràmetres per a la línia d'ordres, i un entorn d'execució (com «executa aquest programa en un intèrpret d'ordres» o «executa aquest programa en el depurador»).

6.1 Configurar els llançaments al KDevelop



Per a configurar-los, aneu a l'element de menú **Executa** → **Configura els llançaments**, ressalteu el projecte pel qual voleu afegir un llançament i feu clic al botó . A continuació, introduïu el nom de l'executable i el camí on es troba el programa que voleu executar. Si la seva execució depèn de la construcció d'altres executables i/o biblioteques, llavors és probable que el vulgueu afegir a la part inferior de la llista: seleccioneu **Construeix** des del menú desplegable, després

premeu el símbol a la dreta del quadre de text i seleccioneu qualsevol objectiu que vulgueu construir. En l'anterior exemple, he seleccionat l'objectiu **tot** del projecte `1.deal.II` i `step-32` del projecte `1.step-32` per a assegurar-me que tant la biblioteca base com el programa de l'aplicació s'han tornat a compilar i estan al dia abans d'executar realment el programa. Quan hi arribeu, és possible que també vulgueu configurar un llançament per a la depuració, feu clic al símbol **Depuració** i afegiu el nom del programa depurador. Si aquest és el depurador per omissió del sistema (p. ex., el gdb a Linux®), llavors no caldrà que realitzeu aquest pas.



Ara podeu provar d'executar el programa: Seleccioneu **Executa** → **Executa el llançament** des del menú a la finestra principal del KDevelop (o premeu **Maj-F9**) i el vostre programa s'hauria d'executar en una finestra secundària separada del KDevelop. La imatge de dalt mostra el resultat: La subfinestra nova de l'eina **Executa** a la part inferior mostra la sortida del programa que s'està executant, en aquest cas del programa `step-32`.

> NOTA
> Si heu configurat múltiples llançaments, podeu triar quin s'haurà d'executar prement **Maj-F9** i anant a **Executa** → **Configuració de llançament actual**. No obstant això, existeix una manera no evident per a editar el nom d'una configuració: en el diàleg que s'obté quan seleccioneu **Executa** → **Configuració de llançament actual**, feu doble clic sobre el nom de la configuració a la vista en arbre de l'esquerra, el qual us permetrà editar el nom de la configuració.

6.2 Algunes dreceres de teclat útils

Executar un programa	
F8	Construeix (crida al «make»).
Maj-F9	Executa

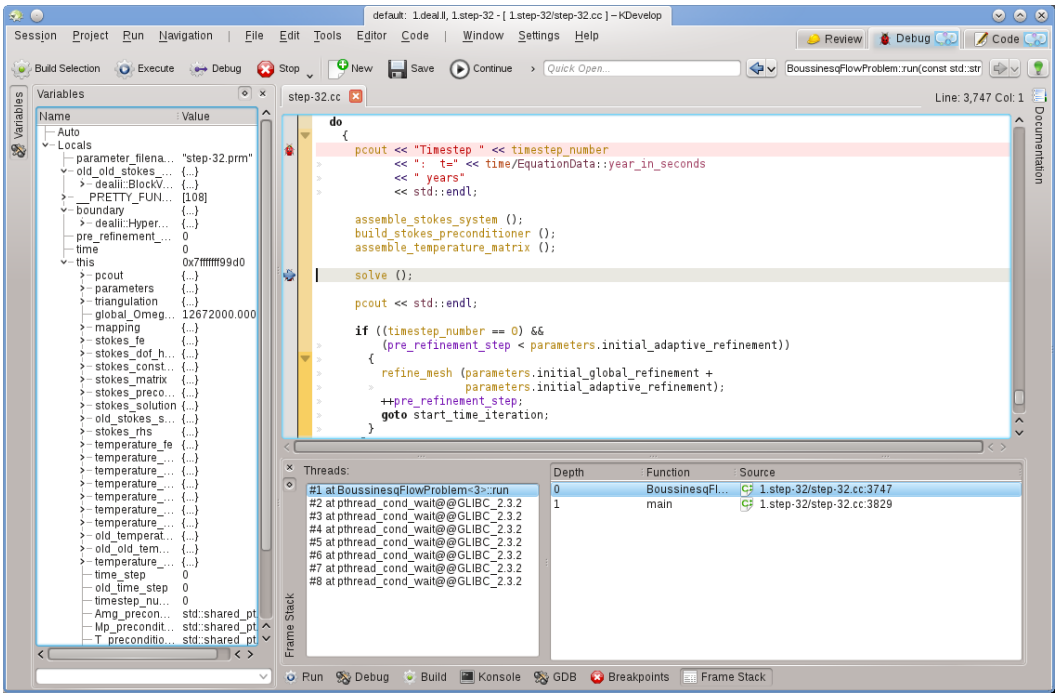
Alt-F9	Executa el programa en el depurador. Possiblement abans voldreu establir punts d'interrupció, p. ex., fent clic dret amb el ratolí sobre una determinada línia en el codi font.

Capítol 7

Depurar programes al KDevelop

7.1 Executar un programa en el depurador

Una vegada tingueu configurat un llançament (vegeu Executar programes), també el podeu executar en un depurador: Seleccioneu l'element de menú **Executa** → **Llança al depurador** o premeu **Alt-F9**. Si esteu familiaritzat amb el gdb, l'efecte és el mateix que iniciar el gdb amb l'executable especificat en la configuració del llançament i després dient `Executa`. Això vol dir que si el programa crida `abort()` en algun moment (p. ex., quan s'executa sobre una asserció que falla) o si hi ha un error de segmentació, a continuació s'aturarà el depurador. D'altra banda, si el programa s'executa fins al final (de forma correcta o no), llavors el depurador no s'aturarà per si mateix fins que finalitzi el programa. En aquest últim cas, haureu d'establir un punt d'interrupció en totes les línies del codi base en les quals voleu que el depurador s'aturi abans d'executar el llançament de depuració. Ho podeu fer movent el cursor a una certa línia i seleccionant l'element de menú **Executa** → **Commuta el punt d'interrupció**, o fent clic dret en una línia i seleccionant **Commuta el punt d'interrupció** des del menú contextual.

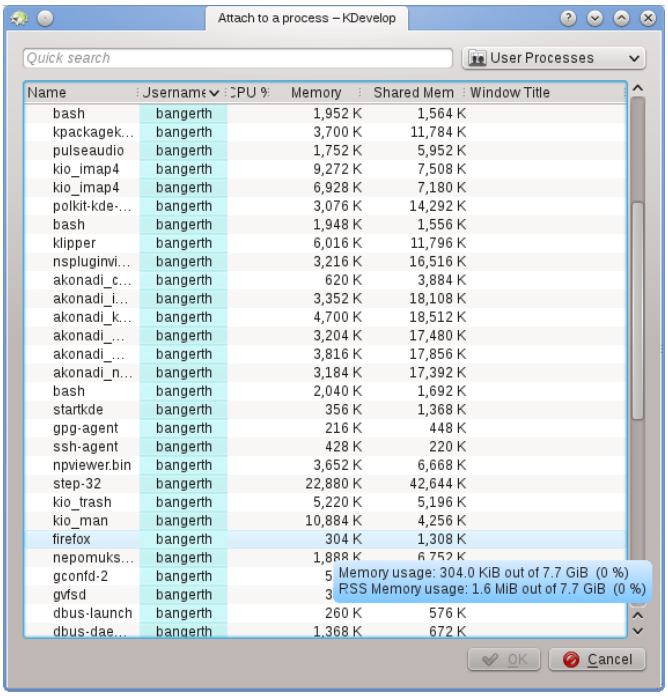


Executar un programa en el depurador posarà el KDevelop en un mode diferent: se substituiran tots els botons d'«eina» en el perímetre de la finestra principal per altres que siguin apropiats per a la depuració, en lloc de per als d'edició. Podeu veure en quin mode es troba observant la part superior dreta de la finestra: hi ha pestanyes anomenades **Revisió**, **Depuració** i **Codi**. En fer clic en elles se us permetrà alternar entre els tres modes. Cada mode té un conjunt de vistes d'eina pròpia, que es poden configurar de la mateixa manera que hem configurat les eines **Codi** a la secció Eines i vistes.

Una vegada s'atura el depurador (en un punt d'interrupció o punt on es crida `abort()`), podreu inspeccionar una varietat d'informació sobre el vostre programa. Per exemple, en la imatge anterior, hem seleccionat l'eina **Pila d'execució** a la part inferior (més o menys equivalent a les ordres «backtrace» i «info threads» del gdb) que mostra a l'esquerra els diversos fils que s'estan executant actualment en el vostre programa (aquí un total de 8) i com l'execució arriba al punt d'aturada actual a la dreta (aquí: «main()» ha cridat a «run()»; la llista podria ser més llarga de no haver-se aturat a una funció anomenada `run()`). A l'esquerra, podem inspeccionar les variables locals, inclòs l'objecte actual (l'objecte apuntat per la variable `this`).

A partir d'aquí, hi ha diverses possibilitats: Podeu executar la línia actual (**F10**, l'ordre «next» del gdb), entrar en les funcions (**F11**, l'ordre «step» del gdb) o executar fins al final de la funció (**F12**, l'ordre «finish» del gdb). A cada pas, el KDevelop actualitza les variables que es mostren a l'esquerra als seus valors actuals. També podeu passa el ratolí sobre un símbol en el codi, p. ex., una variable, el KDevelop mostrarà el valor actual d'aquest símbol i oferirà aturar el programa durant l'execució la pròxima vegada que es canviï el valor d'aquesta variable. Si coneixeu el gdb, també podeu fer clic al botó de l'eina **GDB** a la part inferior i tenir la possibilitat d'introduir ordres del gdb, p. ex., amb la finalitat de canviar el valor d'una variable (per al qual actualment no sembla haver-hi una altra manera).

7.2 Adjuntar el depurador a un procés en execució



A vegades, un vol depurar un programa que ja s'està executant. Un escenari per a això és la depuració de programes paral·lels utilitzant MPI, o per a la depuració d'un procés en segon pla de llarga durada. Per a fer-ho, aneu a l'element de menú **Executa** → **Adjunta al procés**, el qual

obrirà una finestra com l'anterior. Haureu de seleccionar el programa que s'adapti al vostre projecte actualment obert al KDevelop -en el meu cas seria el programa step-32.

Aquesta llista de programes pot resultar confusa, perquè sovint és molt llarga, com en el cas que es mostra aquí. Podeu fer que les coses siguin una mica més fàcils anant a la llista desplegable a la part superior dreta de la finestra. El valor per omissió és **Processos dels usuaris**, és a dir, tots els programes que s'executen per qualsevol dels usuaris actualment connectats a aquesta màquina (si es tracta del vostre ordinador o portàtil, és probable que sigui un únic usuari, a part de l'arrel i els diversos comptes de servei). La llista no inclou els processos executats per l'usuari «root». Podeu limitar la llista escollint **Processos propis**, eliminant tots els programes executats pels altres usuaris. O millor encara: Seleccioneu **Només els programes**, el qual elimina una gran quantitat de processos que s'executen formalment sota el vostre nom, però que no solen interactuar, p. ex., el gestor de finestres, tasques en segon pla i així successivament, són candidats improbables per a la depuració.

Un cop tingueu seleccionat un procés, que s'associï amb ell entrareu en el mode de depuració del KDevelop, obrirà totes les vistes d'eina de depuració habituals i aturarà el programa a la posició en la qual el vau associar a ell. Possiblement, després voldreu establir punts d'interrupció, punts de vista o qualsevol altra cosa que sigui necessària i continuar amb l'execució del programa, aneu a l'element de menú **Executa** → **Continua**.

7.3 Algunes dreceres de teclat útils

Depurar	
F10	Avança sobre (l'ordre «next» del gdb).
F11	Avança dins (l'ordre «step» del gdb).
F12	Avança fora de (l'ordre «finish» del gdb).

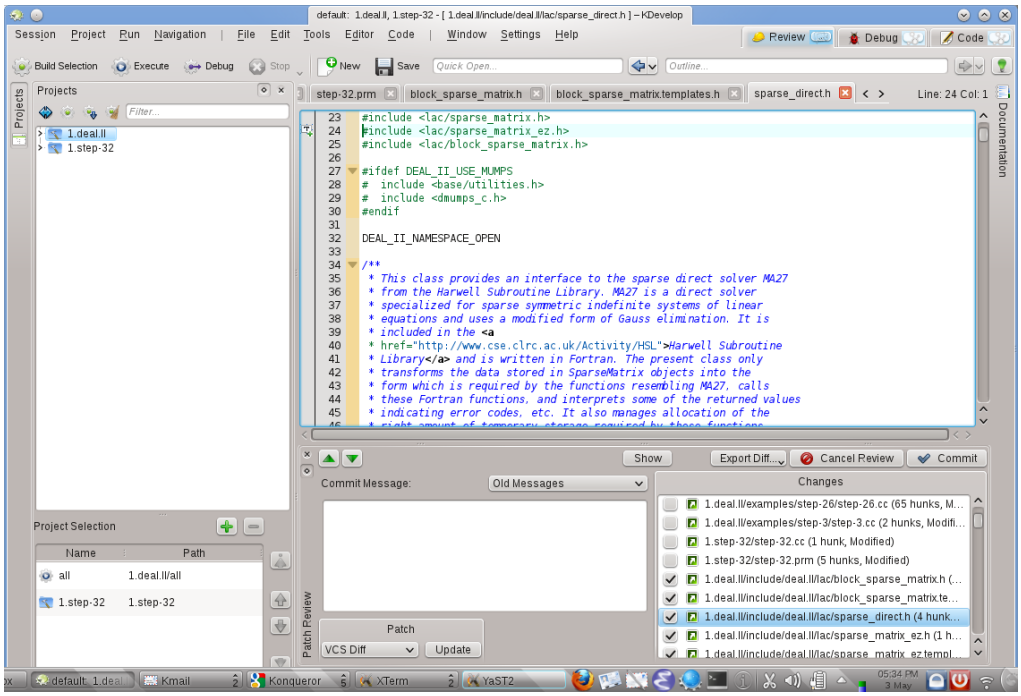
Capítol 8

Treballar amb sistemes de control de versions

Si esteu treballant amb projectes molt grans, el més probable és que el codi font sigui gestionat per un sistema per al control de versions com subversion o git. La següent descripció s'escriu amb **subversion** a la vista, però també és certa si empreu **git** o qualsevol altre sistema admès per al control de versions.

En primer lloc, si el directori en el qual es troba un projecte està sota el control de versions, el KDevelop s'adonarà de forma automàtica. En altres paraules: No cal indicar-li que el baixi per a configurar el vostre projecte. Només cal apuntar el KDevelop a un directori en el qual heu baixat prèviament una còpia del repositori. Si teniu un directori sota el control de versions, obriu la vista d'eina **Projectes**. Doncs hi ha una sèrie de coses que podeu fer:

- Si el directori s'ha convertit en obsolet, podeu actualitzar-lo des del repositori: Feu clic al nom del projecte amb el botó dret del ratolí, aneu al menú **Subversion** i seleccioneu **Actualitza**. Això farà que tots els fitxers que pertanyen a aquest projecte siguin actualitzats amb els del repositori.
- Si voleu restringir aquesta acció a subdirectoris o fitxers individuals, a continuació expandiu la vista en arbre d'aquest projecte al nivell que vulgueu i feu clic dret sobre el nom del subdirectori o fitxer, i després fer el mateix que anteriorment.



- Si heu editat un o més fitxers, expandiu la vista d'aquest projecte al directori en el qual es troben aquests fitxers i feu clic dret sobre el directori. Això us oferirà un element de menú **Subversion** que us oferirà diferents opcions. Trieu **Compara amb la base** per a veure les diferències entre la versió que heu editat i la versió en el repositori que havíeu actualitzat prèviament (la «base» de la revisió). La vista resultant mostrarà els «diffs» per a tots els fitxers en aquest directori.

- Si només editeu un sol fitxer, també podeu obtenir el menú **Subversion** per aquest fitxer, simplement fent clic dret al nom del fitxer corresponent a la vista del projecte. Encara més simple, feu clic dret a la vista **Editor** on heu obert aquest fitxer i també us donarà aquesta opció de menú.

- Si voleu entregar un o més fitxers editats, feu clic dret en un fitxer individual, subdirectori, o tot el projecte i seleccioneu **Subversion** → **Comet**. Això us aconseguirà el mode **Revisió**, el tercer mode a més de **Codi** i **Depuració** com es pot veure a la cantonada superior dreta de la finestra principal del KDevelop. La imatge de la dreta us ho mostra. En el mode **Revisió**, la part superior mostra els diffs per al subdirectori sencer/projecte i cada fitxer canviat amb els canvis ressaltats (vegeu les diferents pestanyes en aquesta part de la finestra). Per omissió, tots els fitxers amb canvis pertanyen al conjunt de canvis que esteu a punt de cometre, però es poden desseleccionar alguns fitxers si les seves modificacions no estan relacionades amb el que voleu cometre. Per exemple, a l'exemple de la dreta no tinc seleccionat `step-32.cc` i `step-32.prm` pel fet que els canvis en aquests fitxers no tenen res a veure amb els altres que vaig fer per a aquest projecte i encara no vull cometre'ls (posteriorment puc voler fer-ho en una altra comissió). Després de revisar els canvis podeu introduir un missatge de comissió en el quadre de text i prémer **Comet** a la dreta per a enviar-ho tot.

- Igual que amb visualitzar les diferències, si voleu entregar un sol fitxer, també podeu fer clic dret a la finestra de l'editor per a obtenir l'element de menú **Subversion** → **Comet**.

Capítol 9

Personalitzar el KDevelop

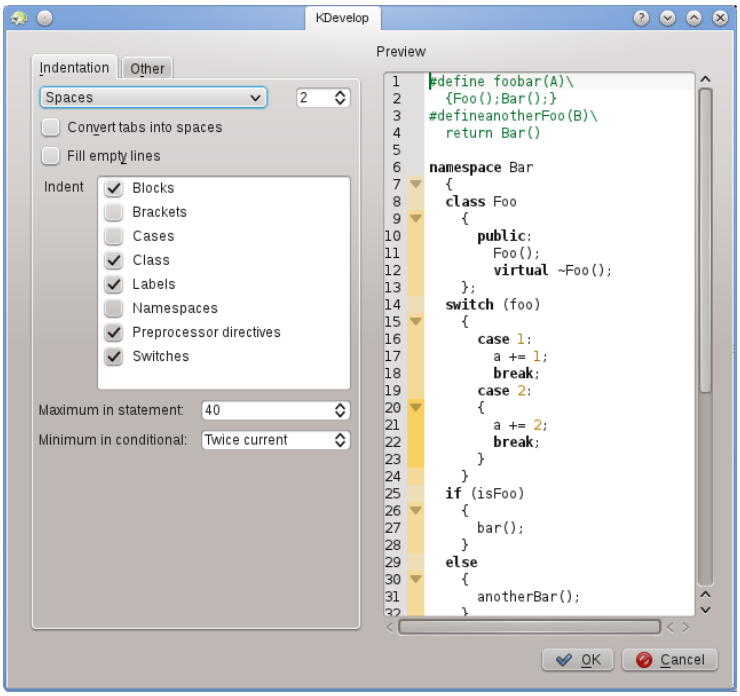
Hi ha moments en què hom vol canviar l'aparença o el comportament per omissió del KDevelop, p. ex., perquè us heu acostumat a diferents dreceres de teclat o perquè el vostre projecte requereix un estil de sagnat diferent per al codi font. En les següents seccions, es discuteixen breument les diverses maneres com es pot personalitzar el KDevelop per a aquestes necessitats.

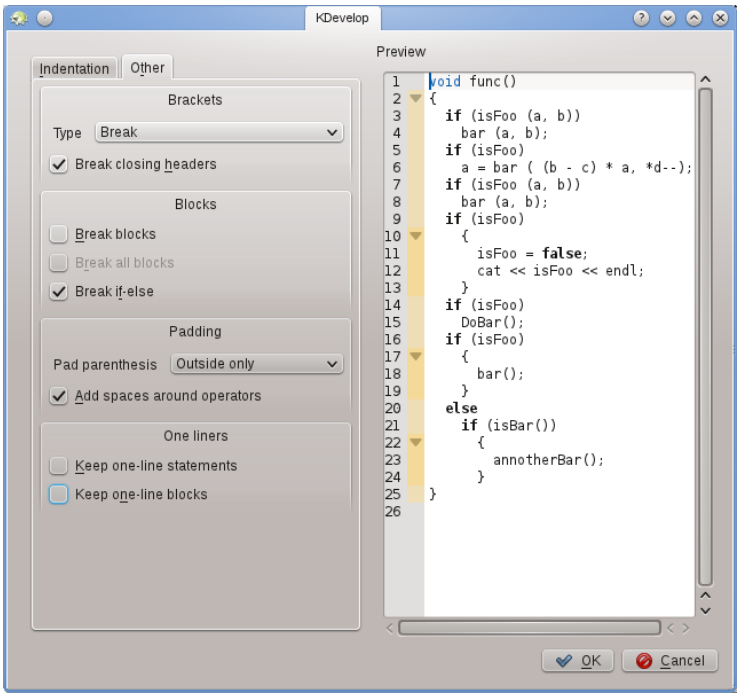
9.1 Personalitzar l'editor

Hi ha un nombre de coses útils que es poden configurar relacionades amb l'editor integrat del KDevelop. Una de les més universals consisteix a canviar la numeració de les línies emprant l'entrada de menú **Editor** → **Visualitza** → **Mostra els números de les línies**, de manera que és més fàcil que coincideixin els missatges d'error del compilador o missatges de depuració amb les ubicacions en el codi. En el mateix submenú és possible que també vulgueu canviar la *Vora de la icona* -una columna a l'esquerra del vostre codi en el qual el KDevelop mostrarà icones com p. ex., si hi ha un punt d'interrupció en la línia actual-.

9.2 Personalitzar el sagnat del codi

Molts de nosaltres ens agrada tenir el codi amb format d'una manera particular. Molts projectes també imposen un estil de sagnat en particular. Tampoc té per què coincidir amb l'estil de sagnat per omissió del KDevelop. No obstant això, es pot personalitzar: Aneu a l'element de menú **Arranjament** → **Personalitza el KDevelop**, després feu clic a **Formatador del codi font** de l'esquerra. Podeu triar un dels estils de sagnat predefinits que estan àmpliament en ús, o definir el vostre propi afegint un estil nou i després editant-lo. Pot ser que no hi hagi una manera de tornar a crear exactament l'estil en què s'han sagnat les fonts del vostre projecte en el passat, però us hi podeu apropar emprant la configuració d'un estil nou. A les dues fotos de sota es mostra un exemple.


KDevelop
Indentation
Other
Spaces
2
Convert tabs into spaces
Fill empty lines
Indent
Blocks
Brackets
Cases
Class
Labels
Namespaces
Preprocessor directives
Switches
Maximum in statement:
40
Minimum in conditional:
Twice current
Preview
OK
Cancel


KDevelop
Indentation
Other
Brackets
Type
Break
Break closing headers
Blocks
Break blocks
Break all blocks
Break if-else
Padding
Pad parenthesis
Outside only
Add spaces around operators
One liners
Keep one-line statements
Keep one-line blocks
Preview
OK
Cancel

NOTA
Amb el **KDevelop 4.2.2**, podeu crear un estil nou per a un tipus MIME en particular (p. ex., pels fitxers de capçalera de C++), però aquest estil no es mostrarà a la llista de possibles estils per a altres tipus MIME (p. ex., pels fitxers de codi font de C++) tot i que es dona per fet que seria útil emprar el mateix estil per a ambdós tipus de fitxers. Per tant, haureu de definir l'estil dues vegades, un per a les capçaleres i un altre pels fitxers d'origen. Això ha estat informat reportat com a l'error 272335 del KDevelop.

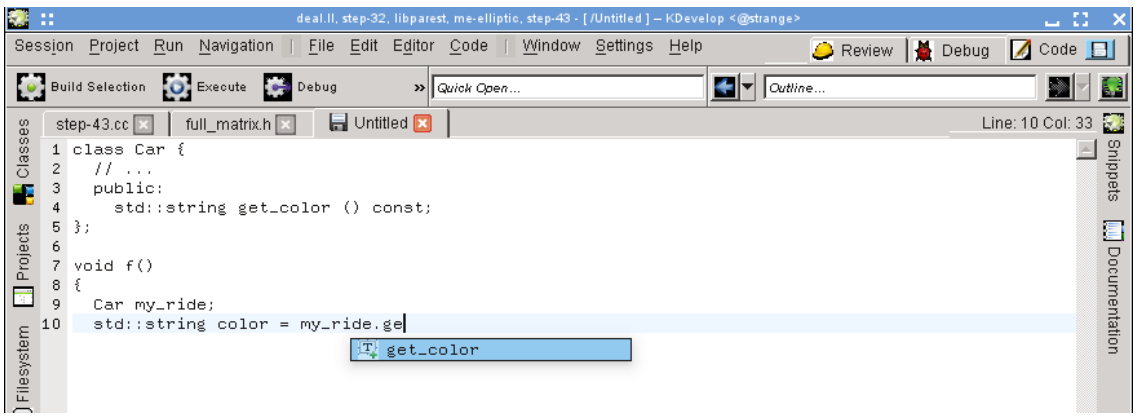
9.3 Personalitzar les dreceres del teclat

El KDevelop disposa d'una llista gairebé il·limitada de dreceres de teclat (algunes d'elles s'enumeren en les «seccions per a dreceres de teclat útils» de diversos capítols en aquest manual) que es poden canviar al vostre gust mitjançant el menú **Arranjament** → **Configura les dreceres**. A la part superior del diàleg podeu introduir una paraula de cerca i només es mostraran les ordres que coincideixin. Després podreu editar la combinació de tecles que s'associïn amb aquesta ordre.

Dues que he trobat molt útils per a canviar són establir **Alinea** a la tecla **Tab** (moltes persones no solen introduir les tabulacions a mà i més aviat prefereixen que l'editor decideixi la disposició del codi, amb la drecera canviada, el fet de prémer **Tab** farà que el KDevelop sagni/anul·li el sagnat/alineï el codi). La segona és posar **Commuta el punt d'interrupció** a **Ctrl-B** atès que es tracta d'una operació molt freqüent.

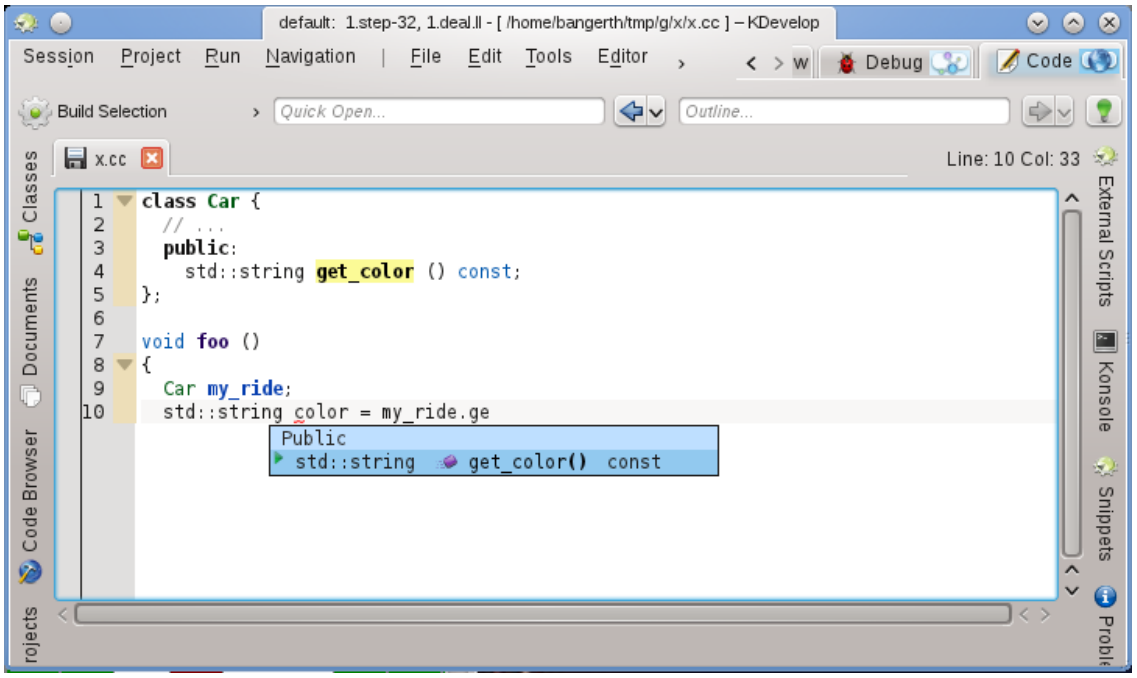
9.4 Personalitzar la compleció automàtica del codi

La compleció del codi es discuteix a la secció d'aquest manual sobre com escriure codi font. En el KDevelop, prové de dues fonts: l'editor i el motor d'anàlisi. L'editor (Kate) és un component del gran entorn KDE i ofereix la compleció automàtica basant-se en paraules que ja ha vist en altres parts del mateix document. Aquestes complecions automàtiques es poden identificar en el consell per la icona que el precedeix:



La compleció de codi de l'editor es pot personalitzar a través d'**Arranjament** → **Configura l'editor** → **Edició** → **Compleció automàtica**. En particular, podeu seleccionar el nombre de caràcters que cal teclejar d'una paraula abans que es mostri la compleció automàtica.

D'altra banda, la mateixa compleció automàtica del KDevelop és molt més potent, ja que té en compte informació semàntica sobre el context. Per exemple, sap quines funcions de membre oferir en escriure `object.`, etc., com es mostra aquí:



Aquesta informació de context prové de diversos connectors d'implementació del llenguatge, que es poden utilitzar després de desar un determinat fitxer (així que podeu comprovar el tipus de fitxer i emprar la implementació pel llenguatge correcte).

La compleció del KDevelop està preparada per a aparèixer a mesura que escriviu, immediatament, gairebé a tot arreu on possiblement es pot completar quelcom. Això es pot configurar a **Configuració** → **Configura el KDevelop...** → **Implementació del llenguatge**. Si no s'ha establert ja (com hauria d'estar, per omissió), assegureu-vos que s'estableix **Activa la invocació automàtica**.

El KDevelop té dues maneres de mostrar una compleció: La **Compleció mínima automàtica** només mostra la informació bàsica en els consells de compleció (és a dir, l'espai de nom, classe, funció o nom de la variable). Aquesta serà similar a la compleció del Kate (a excepció de les icones).

D'altra banda, la **Compleció completa** mostrarà a més el tipus de cada entrada, i en el cas de les funcions, també els arguments que prenen. A més, si esteu omplint els arguments d'una funció, la compleció completa tindrà un rètol informatiu addicional sobre el cursor que us mostrarà l'argument actual en què esteu treballant.

La compleció de codi del KDevelop també ha de mostrar a dalt i ressaltar en verd qualsevol element de compleció que coincideixi amb el tipus actualment esperat tant en la compleció mínima com en la completa, coneguda com a «millors coincidències».

Les tres opcions possibles per al nivell de compleció en el diàleg de configuració són:

- **Sempre la compleció mínima**: Mai mostrarà la «Compleció completa».
- **Compleció automàtica mínima**: Només mostrarà la «Compleció completa» quan la compleció automàtica s'hagi activat manualment (és a dir, cada vegada que premeu **Ctrl-Espai**).
- **Sempre la compleció completa**: Mostra sempre la «Compleció completa».

Capítol 10

Crèdits i llicència

Copyright de la documentació, vegeu la pàgina historial «KDevelop4/Manual» d'UserBase.

Traductor de la documentació: Antoni Bella antonibella5@yahoo.com

Aquesta documentació està llicenciada d'acord amb les clàusules de la Llicència de Documentació Lliure de GNU.