Cơ sở lí thuyết CAN database

2.1 Cơ sở dữ liệu hệ thống CAN

2.1.1 Cơ sở dữ liệu.

**Định nghĩa dữ liệu:**

Dữ liệu là các thông tin của đối tượng, chúng ta có thể truy xuất vào dữ liệu để lấy thông tin.

Dữ liệu có thể được mô tả dưới nhiều dạng khác nhau: các kí tự, hình ảnh, số liệu, âm thanh…ứng với mỗi cách mô tả là một ngữ nghĩa khác nhau.

**Định nghĩa cơ sở dữ liệu:**

Cơ sở dữ liệu (database) là một tập hợp dữ liệu được tổ chức có cấu trúc liên quan với nhau và được lưu trữ trong máy tính.

Cơ sở dữ liệu được thiết kế, xây dựng, cho phép người dùng lưu trữ dữ liệu, truy xuất thông tin hoặc cập nhật dữ liệu.

**Các đặc trưng cơ bản của cơ sở dữ liệu:**

2.1.2 Cơ sở dữ liệu CAN

Cơ sở dữ liệu CAN là cơ sở dữ liệu khá đơn giản, nơi mà thông tin được lưu trữ dưới dạng văn bản thuần.

Dữ liệu trong cơ sở dữ liệu CAN sẽ chứa thông tin mô tả cấu trúc hệ thống CAN(Controller Network Area). Các thông tin đó mô tả một hệ thống CAN hoàn chỉnh có phân cấp rõ ràng:

* Trong mạng sẽ chứa các danh sách các node mạng và danh sách các Signal/Message trong các node đó. Hệ thống hoạt động theo đơn vị dữ liệu là các message(có thể chứa 1 hoặc nhiều signal)
* Danh sách các ECUs(Electronic Controll Unit) chứa các biến môi trường(EVs) và các node mạng
* Trong mỗi node mạng sẽ phân biệt 2 loại message: Tx message và Rx message tương ứng với thông điệp phát đi và nhân được từ mỗi ECU.
* Một danh sách các message sẽ lưu thông tin các message của toàn bộ hệt thống, có thể đã được dùng hoặc chưa được dùng. Mỗi message sẽ chứa danh sách các signal trong message đó. Mỗi message có thể có 1 hoặc nhiều signal, tuy nhiên dữ liệu mà các signal chứa không được vượt quá dữ liệu mà message chứa. Một message không chứa signal sẽ không có ý nghĩa trong hệ thống (thường là các message vứa được tạo mới).
* Một danh sách các signal sẽ lưu thông tin tất cả các signal, kể cả những signal được tạo ra nhưng chưa sử dụng trong bất kì message nào.

2.3 Các phương pháp lập trình:

2.3.1. Lập trình tuyến tính.

Đặc trưng cơ bản của lập trình tuyến tính là tư duy theo lối tuần tự. Chương trình sẽ được thực hiện từ đầu đến cuối, lệnh này kế tiếp lệnh kia cho đến hết chương trình.

Đăc trưng:

- Đơn giản: chương trình được thực hiện theo lối tuần tự, không phức tạp

- Đơn luồng: chỉ có một việc công việc duy nhất, và các công việc được thực hiện trong luồng đó

Tính chất:

-Ưu điểm: do tính chất đơn giản, dễ hiểu, lập trình tuyến tính được ứng dụng trong các chương tình đơn giản.

- Nhược điểm: không thể ứng dụng lập trình tuyến tính trong các ứng dụng lớn. Ngày nay, lập trình tuyến tính chỉ còn tồn tại trong các module nhỏ nhất của các phương pháp lạp trình khác.

2.3.2. Lập trình hướng thủ tục(hướng cấu trúc):

Phương pháp **lập trình thủ tục** (*procedural programming*) chia một chương trình (chức năng) lớn thành các khối chức năng hay hàm (thủ tục) đủ nhỏ để dễ lập trình và kiểm tra. Mỗi hàm có một điểm bắt đầu và một điểm kết thúc và có dữ liệu và logic riêng. Trong một hệ thống chương trình, các biến có các phạm vi nhìn thấy nhất định. Trong chương trình, các hàm làm việc độc lập với nhau. Dữ liệu được chuyển đổi qua lại thông qua các tham số gọi hàm. Việc chia chương trình thành các hàm cho phép nhiều người có thể tham gia vào việc xây dựng chương trình. Mỗi người xây dựng một hay một số các hàm độc lập với nhau. Phương pháp này dẫn đến một khái niệm mới – sự trừu tượng hóa. Sự trừu tượng hóa có thể xem như khả năng quan sát một sự việc mà không cần xem xét đến các chi tiết bên trong của nó. Trong một chương trình thủ tục, chúng ta chỉ cần biết một hàm nào đó có thể làm được những công việc cụ thể gì là đủ. Còn làm thế nào để thực hiện công việc đó là không quan trọng, chừng nào hàm còn tin cậy được thì còn có thể dùng nó mà không cần phải biết nó thực hiện đúng đắn chức năng của mình như thế nào. Điều này gọi là sự trừu tượng hóa theo chức năng (*functional abstraction*) (hay còn gọi là sự chuyên môn hóa) và là nền tảng của lập trình thủ tục.

Đặc trưng: Chương trình chính được chia nhỏ thành các chương trình con, mỗi chương trình con thực hiện một công việc xác định.

Chương trình chính sẽ gọi chương trình con theo một cấu trúc hoặc một giải thuật xác định trong chương trình chính.

Đặc trưng:

Đặc trưng cơ bản nhất của lập trình hướng cấu trúc thể hiện ở mối quan hệ:

Chương trình = Cấu trúc dữ liệu + Giải thuật

Cấu trúc dữ liệu: là cách tôt chức dữ liệu cho việc xử lí của một hay nhiều chương trình nào đó.

Giải thuật: là một quy trình để thực hiện một công việc cụ thể

Mối quan hệ giữa giải thuật và cấu trúc dữ liệu:

Một cấu trúc dữ liệu chỉ phù hợp với một số lượng hạn chế các giải thuật

Nếu thay đổi cấu trúc dữ liệu thì phải thay đổi giải thuật cho phù hợp

Một giải thuật thường phải đi kèm với một cấu trúc dữ liệu nhất định

Tính chất:

Mỗi chương trình con có thể được thực hiện nhiều lần trong chương trình chính

Các chương trình con có thể được gọi trong chương trình chính để thực hiện theo một thứ tự bất kì, tùy thuộc vào giải thuật mà không phụ thuộc vào thứ tự khai báo các chương trình con.

Ưu điểm: chương trình rõ ràng, dễ hiểu, dễ theo dõi

Tư duy giải thuất rõ ràng

Nhược điểm:

Lập trình cấu trúc không hổ trợ mạnh mẽ việc sử dụng lại mã nguồn. Giải thuật luôn phụ thuộc chặt chẽ vào cấu trúc dữ liệu, do đó khi thay đổi cấu trúc dữ liệu phải thay đổi giải thuật, nghĩa là viết lại chương trình.

Không phù hợp với phần mềm lớn: tư duy cấu trúc chỉ phù hợp với các bài toán nhỏ, nằm trong phạm vi một module của chương trình.Với các dự án phần mềm lớn, lập trình hướng cấu trúc tỏ ra không hiệu qủa.

2.3.3: Lập trình hướng đối tượng:

Trong lập trình hướng đối tượng, lập trình viên coi các thực thể trong chương trình là các đối tượng sau đó trưu tượng hóa đối tượng thành lớp đối tượng.

Dữ liệu được tổ chức thành các thuộc tính của lớp. Ta chỉ có thể truy cập dữ liệu thông qua đối tượng và các phương thức của đối tượng

Tính chất:

Tính kế thừa: Chúng ta có thể kế thừa các lớp để xây dựng các lớp dẫn xuất, lớp được kế thừa được gọi là lớp cơ sở. Lớp dẫn xuất sẽ có tất cả các thuộc tính và phương thức của lớp cơ sở mà nó kế thừa. Như vậy ta không cần phải xây dựng lại một đối tượng mới hoàn toàn.

Tính kế thừa của lập trình hướng đối tượng cho phép một lớp có thể kế thừa từ các lớp cơ sở. Khi đó lớp dẫn xuất sẽ có thể sử dụng những thuộc tính và phương thức của lớp cơ sở như là của mình.Ngoài ra, lớp dẫn xuất còn có thể bổ sung thêm các thuộc tính phương thức mới giúp giảm được khối lượng công việc trong lập trình hướng đối tượng.

C++ còn hổ trợ đa kế thừa, nghĩa là một lớp có thể kế thừa từ nhiều lớp khác nhau giúp kế thừa các thuộc tính, phương thức cần thiết từ nhiều lớp.

Tính đa hình:

Đa hình là tính chất đi kèm với kế thừa, do việc kế thừa, một lớp có thể sử dụng lại các phương thức của lớp khác. Tuy nhiên, nếu cần thiết, lớp dẫn xuất có thể định nghĩa lại phương thức của lớp cơ sở. Đó là sự nạp chồng phương thức trong kế thừa. Nhờ sự nạp chồng phương thức này ta chỉ cần gọi tên phương thức từ đối tượng mà không cần biết đối tượng thuộc lớp nào. Chương trình sẽ tự kiểm tra đối tượng thuộc lớp nào và thực hiện phương thức tương ứng.

Tính trừu tượng: Trừu tượng hóa dữ liệu (Data abstraction) liên quan tới việc chỉ cung cấp thông tin cần thiết tới bên ngoài và ẩn chi tiết cơ sở của chúng, ví dụ: để biểu diễn thông tin cần thiết trong chương trình mà không hiển thị chi tiết về chúng. C++ cung cấp đủ các phương thức public tới bên ngoài để thao tác với tính năng của đối tượng và để thao tác dữ liệu đối tượng mà không cần thực sự biết về cách lớp đó đã được triển khai nội tại.

Tính đóng gói:

Các dữ liệu được đóng gói vào trong đối tượng. Mỗi đối tượng có một phạm vi truy nhập riêng.

Không thể truy nhập đến dữ liệu một cách tự do như lập trình cấu trúc.

Muốn truy nhập vào các thuộc tính được bảo vệ, phải thông qua các đối tượng, nghĩa là phải thông qua các phương thức mà đối tượng đó cung cấp mới có thể truy nhập được dữ liệu đã được bảo vệ.

Ưu điểm:

Không còn nguy cơ dữ liệu bị thay đổi trong chương trình vì dữ liệu đã được đóng gói trong các đối tượng. Việc truy xuất dữ liệu phải thông qua các phương thức được cho phép bởi đối tượng.

Khi thay đổi cấu trúc dữ liệu của đối tượng, không cần thay đổi mã nguồn của các đối tượng khác, mà chỉ cần thay đổi một số thành phần của lớp dẫn xuất. Tránh được việc phải thay đổi quá nhiều mã khi thay đổi cấu trúc dữ liệu.

Có thể sử dụng lại mã nguồn, tiết kiệm đáng kể tài nguyên, thời gian và chi phí. Điều này có được nhờ tính kế thừa.

Phù hợp với các phần mềm lớn, phức tạp.

Hạn chế của lập trình hướng đối tượng C++:

Tính đa kế thừa trong C++ có thể dẫn đến xung đột phương thức nếu 1 lớp kế thwuaf từ 2 lớp khác mà 2 lớp đó lại có các phương thức với các API giống nhau. Lúc đó, đối tượng của lớp dẫn xuất sẽ không biết được đang gọi phương thức của lớp cơ sở nào. Vì vậy lập trình viên cần cẩn trọng không lạm dụng đa kế thừa.

Kết luận: Là ngôn ngữ nữa hướng đối tượng, nữa hướng cấu trúc do kế thừa từ ngôn ngữ C thuần cấu trúc. Vì vậy C++ có cả đặc tính hướng cấu trúc và hướng đối tượng, tùy thuộc vào trường hợp cụ thể mà ta sẽ sử dụng phương pháp cụ thể.

Như vậy, C++ vừa có thể sử dụng cho lập trình phần cứng, tác động sâu vào từng vùng nhớ, vừa có tính chất hướng đối tượng giúp tiết kiệm thời gian và tài nguyên.

2.2 Môi trường phát triển .

2.2.1 Hệ điều hành Linux

**2.2.1.1. Khái niệm:**

****

Hình 2.1: Kiến trúc hệ điều hành Linux

Linux là hệ điều hành (Operating System) máy tính sử dụng mã nguồn mở được biết đến rộng rãi và nhiều người dùng nhất.

**2.2.1.2. Ưu điểm của của hệ điều hành.**

* **Bản quyền:**

Lợi thế của Linux chính là nền tảng mã nguồn mở và hoàn toàn miễn phí.

* **Bảo mật**

Nếu như trên Windows bạn luôn phải chật vật đối mặt với ngày càng nhiều những loại virus, mã độc,… thì bạn lại được an toàn khi sử dụng Linux, bởi vì đơn giản, tất cả những loại phần mền độc hại này đều không thể hoạt động được trên nền tảng này. Công việc của bạn chỉ là xóa khi thấy chúng trong USB hay ổ cứng di động.

* **Linh hoạt**

Tính linh hoạt của Linux được thể hiện ở chỗ nó tương thích được với rất nhiều môi trường. Hiện tại, ngoài Linux dành cho server,máy tính để bàn...nhân Linux (Linux kernel) còn được nhúng vào các thiết bị điều khiển như máy tính palm, robot.....Phạm vi ứng dụng của Linux được xem là rất rộng rãi.

* **Độ an toàn cao**

Trước hết, trong Linux có một cơ cấu phân quyền hết sức rõ ràng. Chỉ có "root"( người dùng tối cao) mới có quyền cài đặt và thay đổi hệ thống. Ngoài ra Linux cũng có cơ chế để một người dùng bình thường có thể chuyển tạm thời chuyển sang quyền "root" để thực hiện một số thao tác. Điều này giúp cho hệ thống có thể chạy ổn định và tránh phải những sai sót dẫn đến đổ vỡ hệ thống. Trong những phiên bản Windows gần đây, cơ chế phân quyền này cũng đã bước đầu được áp dụng, nhưng so với Linux thì vẫn kém chặt chẽ hơn.

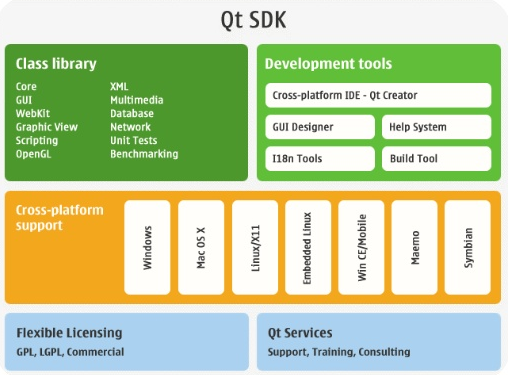
* **Chạy thống nhất trên các hệ thống phần cứng**

Dù cho có rất nhiều phiên bản Linux được các nhà phân phối khác nhau ban hành nhưng nhìn chung đều chạy khá ổn định trên mọi thiết bị phần cứng, từ Intel 486 đến những máy Pentium mới nhất, từ những máy có dung lượng RAM chỉ 4MB đến những máy có cấu hình cực mạnh (tất nhiên là tốc độ sẽ khác nhau nhưng về nguyên tắc vẫn có thể chạy được). Nguyên nhân là Linux được rất nhiều lập trình viên ở nhiều môi trường khác nhau cùng phát triển (không như Windows chỉ do Microsoft phát triển) và bạn sẽ bắt gặp nhiều người có "cùng cảnh ngộ" như mình và dễ dàng tìm được các driver tương ứng với thiết bị của mình. Tính chất này hoàn toàn trái ngược với Windows. Mỗi khi có một phiên bản Windows mới ra đời thì bao giờ kèm theo đó cũng là một cơn khát về phần cứng vì HĐH mới thường không hỗ trợ các thiết bị quá cũ.

2.2.2 QT software framework

**2.2.2.1. Giới thiệu chung về QT Framework:**

* Định nghĩa framework: Là một thư viên các lớp được xây dựng hoàn chỉnh, là nền tảng để phát triển các phần mềm ứng dụng. Thay vì tự tay viết tất cả mã cho các lớp , hàm cho toàn bộ dự án phần mềm của mình, các lập trình viên dùng framework để tiết kiệm thời gian và công sức nhưng vẫn đạt hiệu quả mong muốn bằng cách kế thừa các lớp có sẵn từ các thư viện trong framework



Hình 2.2 Mô hình tổng quan Qt framework

* QT Framework là một nền tảng xây dựng các ứng dụng chạy trên nhiều hệ điều hành khác nhau.
* Mục tiêu của các nhà phát triển nên Qt chính là tạo ra một nền tảng có khả năng thiết kế những phần mềm có thể chạy trên nhiều nền tảng phần mềm lẫn phần cứng khác nhau mà không phải thay đổi nhiều về code.
* Qt hổ trợ các nền tảng sau: Windows, Linux, OS X, iOS, Android, WinRT…
* Qt hổ trợ ngôn ngữ C/C++/Java/Python/…
* Qt hổ trợ lập trình giao diện chuẩn theo mô hình model/view và hổ trợ các lớp lập trình giao diện động trong Qt Quick, giúp tạo giao diện chuyên nghiệp và sinh động hơn.
* Với QT chúng ta không cần thiết phải viết giao diện từ những dòng code phức tạp, mà hoàn toàn có thể vẽ giao diện trong tệp có phần mở rộng .ui như vậy việc thiết kế giao diện dễ dàng hơn nhiều. UI của Qt hổ trợ sẵn các công cụ như QPushButton, QLineEdit, QComboBox, QTreeView, QStackedWidget… giúp việc thiết kế dễ dàng hơn nhiều so với việc phải tự xây dựng tất cả các lớp trên.
* Với những tính năng nổi bật trên, nhóm em quyết định chọn Qt framework làm nền tảng để xây dựng phần mềm CAN database.

**2.2.2.2 Mô hình Model/View trong QT Framework**

* Kiến trúc MVC và kiến trúc model/view trong QT:



Hình 2.3 Kiến trúc mô hình MVC

* Mô hình MVC (Model-View-Controller) là một kiến trúc phần mềm, nó giúp cho các nhà phát triển phần mềm tách các ứng dụng của họ ra thành 3 phần có nhiệm vụ riêng biệt và độc lập với các thành phần khác, trong đó:
* Model: chứa các phương thức xử lí và truy xuất dữ liệu.
* View: Đảm bảo việc hiển thị thông tin và tương tác với người dùng.
* Controller: là trung gian giữa Model và View, nhận các yêu cầu từ tương tác người dùng từ view và tác động làm thay đổi dữ liệu trong model.
* Như vậy, việc xây dựng phần mền dùng MVC sẽ dễ dàng, trực quan hơn so với việc xây dựng theo kiểu truyền thống. Ngoài ra, việc chia thành 3 phần riêng biệt giúp cho việc phát triển, nâng cấp, bảo trì và khắc phụ sự cố tiết kiệm thời gian, công sức và chi phí. Việc chia nhỏ như vậy cuãng thuận tiện để phân chia nhiệm vụ và chuyên biệt hóa các phần trong các dự án lớn.
* QT framework sử dụng mô hình model/view thay cho mô hình MVC, trong đó lớp Controller sẽ được tích hợp trong View thông qua Delegate.

Hình 2.4. Cấu trúc model/view trong QT

* **Lí do lựa chọn mô hình model/view:** mô hình này chia phần mền thành các module riêng biệt, nên sẽ dễ dàng hơn cho việc bảo trì, nâng cấp, phát triển sau này.
* Hoạt động của model/view:
* Ban đầu model sẽ sử dụng phương thức setModelData () của mình để lấy dữ liệu từ tập dữ liệu gốc (dataset), và View sẽ tự cập nhật dữ liệu được lấy từ model thông qua phương thức setModel ().
* Mỗi tương tác chỉnh sửa của người dùng trên giao diện sẽ gọi một đối tượng của 1 lớp Delegate tương ứng, thông qua delegate sẽ làm thay đổi dữ liệu tương ứng trong model, khi đó model sẽ cập nhật hiển thị mới trên View cũng như phát đi một signal để làm thay đổi dữ liệu ở dữ liệu gốc, signal này được kết nối với một slot tương ứng ở lớp quản lí tập dữ liệu gốc, slot này sẽ làm thay đổi dữ liệu trong dataset.

**2.2.2.3 Cơ chế Signal và Slot trong QT Framework**

* Signal và Slot trong QT:
* Trong các phần mềm mà người dùng có thể tương tác được với giao diện, phải có một cơ chế nào đó để lúc người dùng tương tác( như kích chuột, di chuyển con trỏ chuột,…) với giao diện thì phần mềm sẽ biết được người dùng đang thực hiện thao tác đó. Qt sử dụng cơ chế signal/slot để đảm nhận nhiệm vụ này.
* Signal: khi một sự kiện nào đó xảy ra, một signal sẽ được phát đi, thực ra nó chỉ là một phương thức của một lớp nhưng không có phần thân hàm {}. Các lớp Widget có sẵn trong Qt có rất nhiều signal được định nghĩa sẵn, và chúng ta cũng có thể viết các signal riêng cho các lớp của tự định nghĩa. Signal không có kiểu trả về, kiểu trả về của signal luôn luôn là void.
* Slot:  là một phương thức bình thường của một lớp, các phương thức này sẽ được gọi khi có một signal nào đó được phát đi. Cũng giống như signal, các lớp Widget trong Qt cũng có sẵn rất nhiều slot và chúng ta cũng có thể viết slot cho lớp của riêng chúng ta.
* Connect: Signal và slot được kết nối qua từng đối tượng thông qua phương thức connect.

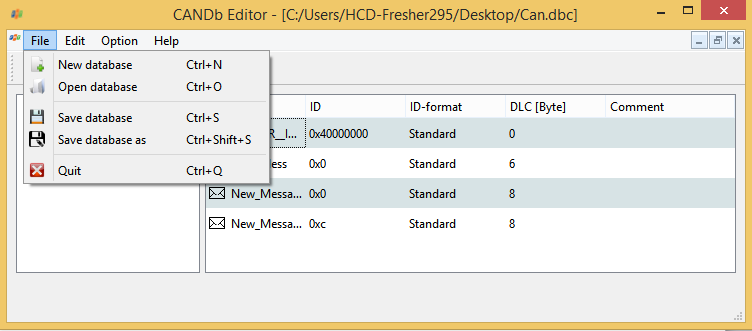
Connect (Đối tượng 1, SIGNAL (signal thuộc đối tượng 1), Đối tượng 2, SLOT (slot thuộc đối tượng 2));

Sau khi kết nối sẽ có một đối tượng phát ra signal và một đối tượng nhận signal đó và thực hiện một phương thức tương ứng, lúc này phương thức đó sẽ là một slot .Kết nối signal và slot là một cách tiện lợi để người dùng gửi đi các signal làm thay đổi dữ liệu tương ứng trên dữ liệu gốc cũng như trên model thay cho delegate.

* Để có thể kết nối signal và slot thì hoặc là phương thức slot của đối tượng nhận signal phải có thành phần tham số giống với signal hoặc là slot phải không có tham số.

2.2.2.4 Các lớp Qt được sử dụng để xây dựng CAN database:

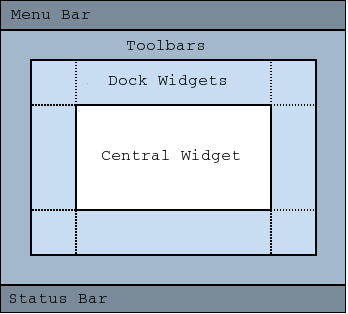
**QWindow:**



Hình 2.2: Cửa sổ giao diện chính sử dụng QWindow và QMenu

* Cửa sổ chính cung cấp một khuôn khổ để xây dựng giao diện cho người dùng.
* QWindow có bố cục riêng mà bạn có thể thêm QToolBars, QMennuBar hay QStatusBar bằng các phương thức setMenuBar (), setStatusbar ()…

**QMainWindow:**



Là cửa sổ chính cung cấp nền tảng cho việc xây dựng một ứng dụng tương tác người dùng. Qt tạo ra QMainWindow và các lớp liên quan để quản lí cửa sổ chính. QMainWindow có layout của chính nó để người dùng có thể thêm QToolBars, QDockWidgets, QMenuBar và QStatusBar.

**QMenu:**



Hình 2.3: Giao diện context menu

Lớp QMenu cung cấp một danh sách các chọn lựa của hành động để dùng trong thanh menu hoặc context menu (menu xuất hiện khi kích chuột vào một vùng nào đó trong QWindow).

**QAction:**

Trong 1 menu chứa một hoặc nhiều QAction, chính là các hành động để người dùng chọn lựa. Khi kích chuột vào một action thì một signal sẽ được giải phóng , tương ứng với nó, 1 slot sẽ được thực thi.

**QMdiArea:**

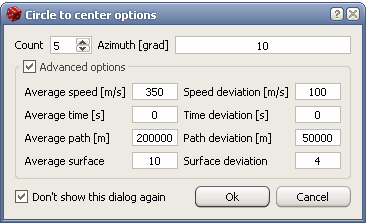
Lớp QMdiArea là tiện ích cung cấp một khu vực trong đó một cửa sổ con được tạo ra trong cưa sổ QMainwindow.

Về cơ bản, giống như cửa sổ quản lý, nó kế thừa hầu hết các chức năng của lớp QMainwindow.



Hình 2.4: Các cửa sổ con được tạo ra trong cửa sổ chính khi gọi lớp QMdiArea

**QDialog:**

****

Hình 2.5: Hộp thoại QDialog

Một hộp thoại tương tự như QWindow nhưng không chứa QMenuBar, QStatusbar, QToolBars.

Hộp thoại QDialog thường là nhưng hộp thoại nhỏ, với các tác vụ đơn giản và tương tác với người dùng trong thời gian ngắn. Ví dụ như các hộp thoại chỉnh sửa dữ liệu, hộp thoại tạo mới đối tượng hay đơn thuần là các hộp thoại hiển thị thông báo, cảnh báo người dùng.

**QFile:**

Lớp QFile cung cấp một giao diện hổ trợ việc đọc và ghi vào tệp.

Là một thiết bị nhập xuất I/O cho việc đọc và ghi tệp dạng văn bản, dạng nhị phân hoặc dạng tài nguyên khác như hình ảnh, âm thanh. QFile có thẻ kết hợp với QTextStream hoặc QDataStream để đọc và ghi dữ liệu từ tệp một cách thuận tiện hơn.

**QStackedWidget:**

**QTableWidget và QTableWidgetItem.**

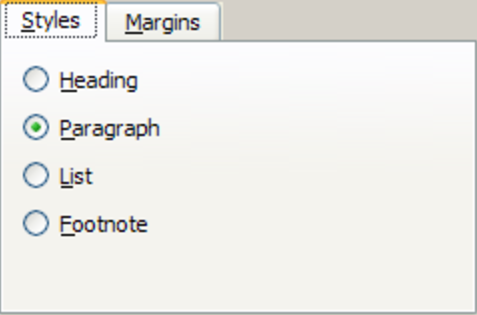
* + - Lớp QTablewidget cung cấp một khung nhìn dựa trên bảng mục tiêu với một mô hình mặc định.
    - Những lớp này cung cấp công cụ xây dựng bảng với số lượng hàng và cột theo yêu cầu.
    - Các tiêu đề trong bảng được tạo ra bằng cách cung cấp một danh sách các chuỗi cho các hàm setHorizontalHeaderLabels () and setVerticalHeaderLabels ().



Hình 2.6: Table được tạo khi gọi lớp QTableWidget, QTableWidgetItem

**QTabWidget:**

QTabWidget cung cấp một ngăn xếp các tab , trong đó mỗi tab là một widget .



**QMessageBox:**

QMessageBox cung cấp một hộp thoại nhằm thông báo cho người dùng hoặc yêu cầu người dùng cung cấp một xác nhận, một chọn lựa cho hành động vứa thực hiện. Ví dụ : Khi xóa 1 thông điệp, do sau khi xóa dữ liệu sẽ mất và không thể khôi phục được nên 1 hộp thoại yêu cầu xác nhận có đúng là người dùng muốn xóa hay không sẽ được hiển thị kèm theo cảnh báo. Hoặc khi thực thi một hành động không hợp lê, sẽ có một cảnh báo tới người dùng thông qua QMessageBox.

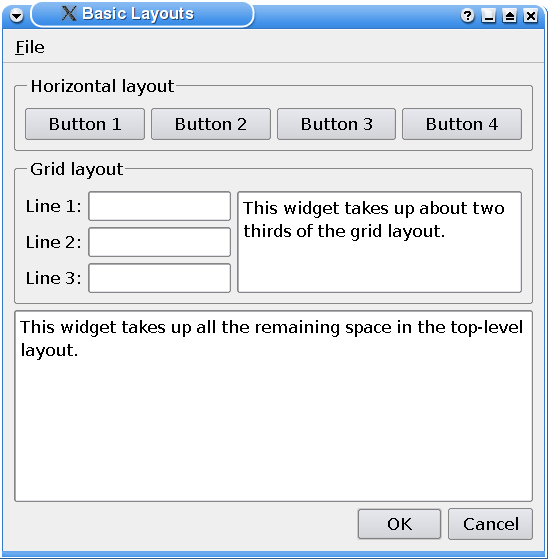
**Layout:**

Qt hổ trợ lớp QLayout cùng các lớp dẫn xuất của nó giúp lập trình viên có thể dễ dang thay đổi định dạng vị trí của các khối chức năng trong các widget, cũng như vị trí của các widget trong các trang.

Layout sẽ khiến giao diện đẹp mắt hơn và thân thiện người dùng hơn. Trong Qt , chúng ta có thể bố trí trang dùng mã hoặc layout bằng giao diện, tức là tự vẽ giao diện layout sau đó chỉnh sửa lại bằng mã C++.

Các lớp dẫn xuất thường dùng của Qt là: **QVBoxlayout, QHBoxLayout, QGridLayout**

* + - Các lớp này xắp xếp các widgets theo một mạng lưới hàng ngang hay hàng dọc, và đặt các widgets mà nó quản lý vào vị trí chính xác.
    - Lớp QVBoxLayout xắp xếp các widgets theo hàng dọc.
    - Lớp QHboxLayout xắp xếp các widgets theo hàng ngang.
    - Lớp QGridLayout tùy chỉnh xắp xếp lại các lớp QVBoxLayout và QHboxLayout bằng cách sử dụng phương thức addlayout ().
    - Khoảng cách giữa các hàng và các cột được tùy chỉnh khi khai báo hàm setColumMinumumWidth ().



Hình 2.7: Các loại Layout trong Qt

Delegate:

Model:

View:

QTreeView

QTableView

QAbstractItemModel

QAbstractItemDelegate

QAbstractItemView

Lập trình GUI (Graphics User Interface) trong QT:

Sử dụng file extension .ui

Tự code hoàn toàn bằng các lớp hổ trợ sẵn của Qt

Layout trong Qt

QMenu

QFile

QTableItem

Action trong Qt