# MỤC LỤC

Contents

[MỤC LỤC 1](#_Toc529087145)

[MỞ ĐẦU 3](#_Toc529087146)

[PHẦN 1 - GIỚI THIỆU TRÌNH BIÊN DỊCH LAZARUS 4](#_Toc529087147)

[PHẦN 2 - GIỚI THIỆU WEBSITE CỦA TRÌNH BIÊN DỊCH LAZARUS 4](#_Toc529087148)

[A. THÔNG TIN CẦN BIẾT 4](#_Toc529087149)

[B. THÔNG TIN QUAN TRỌNG: 5](#_Toc529087150)

[PHẦN 3 - CÀI ĐẶT TRÌNH BIÊN DỊCH LAZARUS 6](#_Toc529087151)

[PHẦN 4 - TRÌNH BIÊN DỊCH LAZARUS VỚI MODE LẬP TRÌNH ĐA DẠNG 7](#_Toc529087152)

[1. Viết code truyền thống - Console-Mode Pascal Programming 7](#_Toc529087153)

[2. Viết code có khai báo thư viện 7](#_Toc529087154)

[3. Viết code lập trình theo hướng đối tượng 8](#_Toc529087155)

[PHẦN 5 - CHƯƠNG TRÌNH LAZARUS ĐẦU TIÊN 9](#_Toc529087156)

[PHẦN 6 - CÁC THƯ VIỆN TRONG FREE PASCAL & LAZARUS 9](#_Toc529087157)

[PHẦN 7 - THƯ VIỆN NGƯỜI DÙNG TỰ TẠO & CÁCH TẠO 9](#_Toc529087158)

[PHẦN 8 - MINH HỌA CÁCH TẠO MỘT THƯ VIÊN 9](#_Toc529087159)

[PHẦN 9 - THỦ THUẬT KHAI THÁC MÃ NGUỒN MỞ TRONG LAZARUS 10](#_Toc529087160)

[1. Danh sách các đối tượng thành phần 10](#_Toc529087161)

[2.Tra cứu mã nguồn mở của các thư viện, thủ tục, hàm … của Trình biên dịch Lazarus 10](#_Toc529087162)

[3. Danh sách thủ tục – Procedure list 11](#_Toc529087163)

[4. Trình quản lý thứ tự của các đối tượng 11](#_Toc529087164)

[5. Làm sạch các mục 11](#_Toc529087165)

[6. Xuất mã nguồn dưới dạng HTML 12](#_Toc529087166)

[7. Những mẫu chương trình ví dụ 12](#_Toc529087167)

[PHẦN 10 - LẬP TRÌNH VỚI ĐỐI TƯỢNG FORM 12](#_Toc529087168)

[1. Công dụng 12](#_Toc529087169)

[2. Tạo ứng dụng Lazarus mới 12](#_Toc529087170)

[3. Lưu ứng dụng mới 13](#_Toc529087171)

[4. Một số thủ thuật cần chú ý : 13](#_Toc529087172)

[PHẦN 11 - CÁC CÁCH THAY ĐỔI THUỘC TÍNH CỦA FORM 14](#_Toc529087173)

[1. Đổi màu FORM 14](#_Toc529087174)

[2. Thay đổi biểu tượng ( Icon ) cho Form 14](#_Toc529087175)

[3. Viết Code cho Form 16](#_Toc529087176)

[16](#_Toc529087177)

[4. Việc biên dịch 16](#_Toc529087178)

[PHẦN 12 PHƯƠNG PHÁP CHUNG THAY ĐỒI THUỘC TÍNH ĐỐI TƯỢNG 16](#_Toc529087179)

[1. Thay đổi trực tiếp trong thẻ Thuộc tính Properties 16](#_Toc529087180)

[2. Thay đổi thuộc tính Form bằng cách viết code 16](#_Toc529087181)

[3. Thay đổi thuộc tính của Form bằng cách di chuyển Form 17](#_Toc529087182)

[PHẦN 13 - LẬP TRÌNH VỚI ĐỐI TƯỢNG NÚT NHẤN - BUTTON 17](#_Toc529087183)

[Vô hiệu hóa Button 17](#_Toc529087184)

[Hiệu lực hóa Button 17](#_Toc529087185)

[Thuộc tính canh chỉnh Button 17](#_Toc529087186)

[Đối tượng SENDER 17](#_Toc529087187)

[PHẦN 14 - LẬP TRÌNH TẠO ỨNG DỤNG CÓ THÊM MỘT FORM THỨ HAI 18](#_Toc529087188)

[PHẦN 15 - LẬP TRÌNH ĐỐI TƯỢNG NHÃN - LABEL 19](#_Toc529087189)

[PHẦN 16 - LẬP TRÌNH VỚI ĐỐI TƯỢNG HỘP VĂN BẢN - EDIT BOX 19](#_Toc529087190)

[PHẦN 17 - LẬP TRÌNH VỚI ĐỐI TƯỢNG HỘP THOẠI - DIALOG 20](#_Toc529087191)

[1. Thủ tục tạo hộp thoại đơn giản - Message Dialogs 20](#_Toc529087192)

[2. Tạo hộp thoại có thể tùy biến - Hàm MessageBox 20](#_Toc529087193)

[3. Hàm MessageDLG – Tạo hộp thông báo phức tạp 22](#_Toc529087194)

[4. Hộp thoại QuestionDlg 23](#_Toc529087195)

[5. Hàm InputBox 23](#_Toc529087196)

[6. Hàm InputQuery 24](#_Toc529087197)

[7. Hàm PasswordBox 25](#_Toc529087198)

[PHẦN 18 - MINH HỌA TẠO ỨNG DỤNG GAME ĐƠN GIẢN 25](#_Toc529087199)

[PHẦN 19 - LẬP TRÌNH VỚI ĐỐI TƯỢNG KHUNG VĂN BẢN MEMO 26](#_Toc529087200)

[PHẦN 20 - MINH HỌA TẠO TRÌNH SOẠN THẢO VĂN BẢN ĐƠN GIẢN 29](#_Toc529087201)

[Chọn Font trong Memo 31](#_Toc529087202)

[Hiển thị ngày giờ trong Memo 31](#_Toc529087203)

[PHẦN 21 - MINH HỌA XỬ LÝ THÔNG TIN TRÊN CÁC THIẾT BỊ LƯU TRỮ 32](#_Toc529087204)

[A. LƯU THÔNG TIN 32](#_Toc529087205)

[B. MỞ THÔNG TIN 33](#_Toc529087206)

[PHẦN 22 - MINH HỌA THAY ĐỔI GIAO DIỆN ỨNG DỤNG 33](#_Toc529087207)

[PHẦN 23 - MINH HỌA TƯƠNG TÁC VỚI TRÌNH DUYỆT & TRANG WEB 34](#_Toc529087208)

[PHẦN 24 - LẬP TRÌNH MÔ PHỎNG THIẾT BỊ ĐÈN TÍN HIỆU GIAO THÔNG 36](#_Toc529087209)

[PHẦN 25 – LẬP TRÌNH MÔ PHỎNG CÁC CHUYỂN ĐỘNG CƠ HỌC 37](#_Toc529087210)

[PHẦN 26 – LẬP TRÌNH MÔ PHỎNG THIẾT BỊ LỊCH ĐIỆN TỬ, ĐỒNG HỒ 39](#_Toc529087211)

[PHẦN 27 - MINH HỌA MENU & ỨNG DỤNG 41](#_Toc529087212)

[PHẦN 28 - MINH HỌA LẬP TRÌNH TẠO HỆ THỐNG MENU NGANG 42](#_Toc529087213)

[A. Menu ngang 42](#_Toc529087214)

[B. Quick menu 43](#_Toc529087215)

[PHẦN 29 - LÀM VIỆC VỚI MỘT DANH SÁCH CHỌN - COMBOBOX 45](#_Toc529087216)

[PHẦN 30 - MINH HỌA LẬP TRÌNH TẠO ỨNG DỤNG DUYỆT HÌNH ẢNH 48](#_Toc529087217)

[PHẦN 31 - MINH HỌA LẬP TRÌNH VẼ ĐƯỜNG TRÊN FORM 48](#_Toc529087218)

[1. Hiệu ứng vẽ đường thẳng nối tiếp 48](#_Toc529087219)

[2. Hiệu ứng pháo bông 49](#_Toc529087220)

[3. Hiệu ứng nhiều chùm pháo bông 49](#_Toc529087221)

[PHẦN 32 - MINH HỌA XỬ LÝ HÌNH HỌC 49](#_Toc529087222)

[PHẦN 33 - BIỂU DIỄN CÁC DẠNG HÌNH HỌC VỚI SỰ KIỆN ONPAINT 50](#_Toc529087223)

[1. Tọa độ vẽ 50](#_Toc529087224)

[2. Vẽ hình, đường bằng Phương thức Canvas 51](#_Toc529087225)

[3. Phương thức Canvas trong sự kiện OnClick 51](#_Toc529087226)

[4. Phương thức Canvas trong sự kiện OnPaint 51](#_Toc529087227)

[PHẦN 34 - MINH HỌA LẬP TRÌNH VẼ HÌNH & TƯƠNG TÁC BẰNG MOUSE 52](#_Toc529087228)

[PHẦN 35 – LẬP TRÌNH ĐỌC & ĐỊNH DẠNG TẬP TIN VĂN BẢN 53](#_Toc529087229)

[PHẦN 36 - LẬP TRÌNH GỌI FILE ỨNG DỤNG & MỞ FILE TÀI LIỆU 55](#_Toc529087230)

[PHẦN 37 - MINH HỌA LẬP TRÌNH TƯƠNG TÁC VỚI CÁC PHÍM MŨI TÊN 55](#_Toc529087231)

[PHẦN 38 – LẬP TRÌNH TỔNG HỢP 56](#_Toc529087232)

## MỞ ĐẦU

**Mục tiêu thực hiện chuyên đề**

* Tăng cường chia sẻ kiến thức, kỹ năng, kinh nghiệm chuyên môn giữa các trường THCS để cùng nhau phát huy năng lực chuyên môn nhằm đáp ứng được yêu cầu giáo dục của bộ môn Tin Học trong giai đoạn hiện nay
* Khai thác tốt nhất tài nguyên mà trang web chuyên mục **Trình biên dịch Lazarus** cung cấp rất chi tiết.
* Xác định những nội dung tham kháo cần đến ở vị trí nào trên trang web để khi cần thiết giúp quý đồng nghiệp nghiên cứu, tham khảo nhanh chóng nhằm tiết kiệm thời gian
* Về kiến thức, kỹ năng sẽ đi vào trong tâm thiết yếu mang tính nền tảng làm cơ sở tiếp tục đổi mới, nâng cao chất lượng, sáng tạo trong học tập và giảng dạy Tin Học trong nhà trường THCS
* Trên thị trường có sách **Tài liệu Dạy – Học Tin học 8** cũng đã đưa vào mảng **Lập trình Hướng đối tượng** thông qua **phần mềm Alice**. Điều này cho thấy những người viết sách Tin học cho học sinh cũng đã quan tâm đến lĩnh vực **Lập trình Hướng đối tượng**, đây chính là xu hướng về tư duy phát triển Công nghệ Tin học hiện nay

Do thời gian eo hẹp rất có thể có những thiếu sót xin quý đồng nghiệp vui lòng bỏ qua

Xin cảm ơn

* Ban giám hiệu Trường THCS Phạm Ngọc Thạch, Thầy Đỗ Quốc Khánh phụ trách Tin học– Phòng GD Tân Bình cùng quý Thầy / Cô tổ Toán – Tin Trường THCS Phạm Ngọc Thạch đã nhiệt tình quan tâm hỗ trợ để giúp tôi thực hiện hoàn tất chuyên đề.
* Các đồng nghiệp từ các Trường THCS trong Quận: Tân Bình đã quan tâm đến với chuyên đề này và tham gia đóng góp ý kiến quý báu

## PHẦN 1 - GIỚI THIỆU TRÌNH BIÊN DỊCH LAZARUS

- Trong website của Trình biên dịch cũng đã giới thiệu rất đầy đủ, nên mục này chủ yếu giúp xác định vị trí trên website để quý đồng nghiệp tham khảo dễ dàng

|  |  |
| --- | --- |
| Click mục **GIỚI THIỆU** |  |
| Tham khảo bài viết |  |
| Để biết được khả năng của **Trình biên dịch Lazarus**, click vào mục |  |
| Lazarus là **Trình biên dịch đa nền tảng** để biết chi tiết, hãy vào mục  , |  |
| Muốn biết các ứng dụng nào đã được tạo ta từ **Trình biên dịch Lazarus** hãy tham khảo mục |  |

## PHẦN 2 - GIỚI THIỆU WEBSITE CỦA TRÌNH BIÊN DỊCH LAZARUS

### A. THÔNG TIN CẦN BIẾT

- Đây là trang website rất quan trọng, những nội dung thiết yếu về kiến thức và kỹ năng giúp đều ở trang này.

- Nhin chung trang web **Trình biên dịch Lazarus** cũng đã đầy đủ những vấn đề cần đề cập nên trong tập tài liệu sẽ không ghi chép lại vào tập tài liệu này. Việc chính là chỉ xác định những vị trí nội dung cần tham khảo trên trang web này giúp quý đồng nghiệp tiết kiệm được thời gian

- **Website** riêng của **Trình biên dịch Lazarus** là [**https://www.lazarus-ide.org/**](https://www.lazarus-ide.org/)có tính chuyên nghiệp cao, tổ chức từng chuyên mục rất khoa học dành cho mọi đối tượng từ người mới bắt đầu cho đến nhà lập trính chuyên nghiệp.

- Đối với người mới bắt đầu có thể bắt gặp điều mình cần ngay trong trang đầu tiên :



|  |  |
| --- | --- |
| - Sau đó trang web tiếp tục dẫn dắt người quan tâm thông qua các đề mục |  |

|  |  |
| --- | --- |
| - Đối với người lập trình chuyên nghiệp cũng có sân cho họ để cống hiến, chia sẻ công trình, sản phẩm của họ qua 3 chuyên mục khác |  |

- Bất cứ ai cũng đều có thể đăng ký tài khoản cho hai chuyên mục **DIỄN ĐÀN** và **WIKI**, mỗi một chuyên mục là một tài khoản riêng.

- Để đăng ký tài khoản

|  |  |
| --- | --- |
| * Click mục **DIỄN ĐÀN** để **Ghi danh** |  |

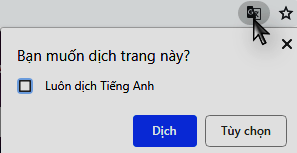
|  |  |
| --- | --- |
| * Click mục **WIKI** để **Tạo tài khoản** |  |

- **Free Pascal** là bản sao của **Turbo Pascal**, còn **Lazarus** là bản sao của **Delphi**. Tuy là bản sao nhưng lại mạnh mẽ hơn so với bản gốc là nhờ vào những nhà lập trình đông đảo hâm mộ vẻ đẹp của Ngôn ngữ Pascal

- Tiếp tục kế thừa **Turbo Pascal** tiếp tục tạo ra những **Bộ thư viện mới** đáp ứng được cả hai hướng:

* **Lập trình theo dòng lệnh** còn gọi là **Console-Mode Pascal Programming**
* **Lập trình hướng đối tượng** còn gọi là **Object Programming**

- Để thuận lợi cho việc nghiên cứu, tham khảo, nên sử dụng **Trình biên dịch Chrome** và dịch vụ **Dịch** để xem tài liệu dạng tiếng Việt



### B. THÔNG TIN QUAN TRỌNG:

- Mục này hướng đến những đối tượng quan trọng trong **Trình biên dịch Lazarus** bao gồm:

* Ý nghĩa, công dụng của từng đối tượng
* Cú pháp viết code cho từng đối tượng

Những trang ví dụ mẫu giúp quý đồng nghiệp có thể tham khảo việc lập trình, triển khai và vận dụng sáng tạo vào công việc giảng dạy bộ môn lập trình Tin học

- Các bước thực hiện:

**Bước 1**:

|  |  |
| --- | --- |
| - Ngay trang chủ của website Lazarus, click nút **Bắt đầu học** |  |

**Bước 2**:

|  |  |
| --- | --- |
| - Trang **Hướng dẫn Lazarus** xuất hiện  - Click chọn **mục** **5 Bảng thành phần**  - Tại mục này giới thiệu đầy đủ các thẻ chứa các đối tượng trên cửa sổ của Trình biên dịch Lazarus  - Hãy thử click vào **thẻ chuẩn Standard**, đây là thẻ chứa các thành phần đối tượng cơ bản nhất |  |

|  |  |
| --- | --- |
|  |  |

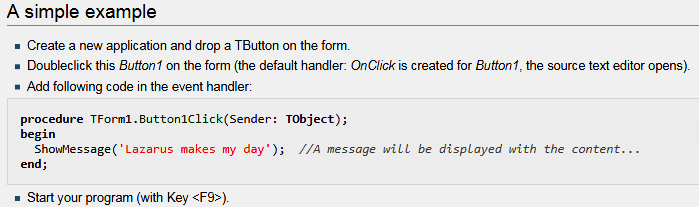
**Bước 3**:

|  |  |
| --- | --- |
| - Chọn thử một đối tượng như **TButton**  - Trong trang này, giới thiệu rất cặn kẽ đối tượng **TButton** này  \* Bước này là bước quan trọng để hiểu rõ đối tượng sẽ phải điều khiển nó |  |

**Bước 4**:

Viết code lập trình đơn giản nhất cho đối tượng **TButton** thông qua một ví dụ đơn giản nhất.

Lần lượt theo 4 điểm là 4 bước để hoàn thành mẫu ví dụ này



**Bước 5**: Sau **Bước 4** tiếp tục tham khảo đối tượng kế tiếp trong thẻ **Standard** và thực hiện tương tự, Khi nắm rõ các đối tượng trong **thẻ Standard**, thì có thể tự tham khảo và thực hiện dễ dàng & bắt đầu tiến bộ rất nhanh

**Chúc quý đồng nghiệp thành công**

## PHẦN 3 - CÀI ĐẶT TRÌNH BIÊN DỊCH LAZARUS

|  |  |
| --- | --- |
| - Chọn phiên bản phù hợp với cấu hình máy tính hiện hành và tải về  - Việc cài đặt diễn ra nhanh và dễ dàng  - Trình biên dịch có tốc độ làm việc hoạt động ổn định ngay cả trên các máy tính chỉ có cấu hình bình thường nên phù hợp với thiết bị máy tính trong nhà trường và không làm nặng máy | C:\Users\LATITUDE\AppData\Local\Temp\SNAGHTML528e2c.PNG |
| - Muốn biết thêm thông tin chi tiết về việc cài đặt, hãy click nút ( Xem hình bên )  -Tiếp tục click đường link **cài đặt** để được hướng dẫn cài đặt với mô tả chi tiết  - Sau đó sẽ đến trang **Cài đặt Lazarus**, trang này hướng dẫn rất chi tiết về việc cài đặt trên những nền tảng hệ điều hành khác nhau |  |
| - Khi **Trình biên dịch Lazarus** hoàn thành cài đặt, có thể vào thư mục **Lazarus > examples** đã cài đặt để nghiên cứu, tham khảo các code ví dụ mẫu |  |

## PHẦN 4 - TRÌNH BIÊN DỊCH LAZARUS VỚI MODE LẬP TRÌNH ĐA DẠNG

- Nhìn chung Trình biên dịch Lazarus đáp ứng được các mode lập trình đa dạng khác nhau

### 1. Viết code truyền thống - Console-Mode Pascal Programming

|  |  |
| --- | --- |
| Vào **File > New > Simple Program**  Xuất hiện cửa Trình biên tập nguồn **Source Editor**  Có thể thấy có sự cải tiến giúp người lập trình giảm bớt tiêu tốn thời gian  Cửa sổ này có ưu điểm: **Copy, Paste, Move …** có dễ dàng hơn trong **Free Pascal** nên thân thiện hơn đối với người lập trình |  |

### 2. Viết code có khai báo thư viện

|  |  |
| --- | --- |
| Vào **File > New > Program**  Xuất hiện Trình biên tập nguồn **Source Editor**  Có thể khai báo thư viện tại vị trí  **{ you can add units after this };**  \* **Diễn giải**:  **- {$mode objfpc}{$H+}** cho các ứng dụng Lazarus  **- {$IFDEF UNIX} …**  Là các dẫn hướng biên dịch mặc định nhằm giúp tránh các xung đột có thể xảy ra do kiến trúc của Lazarus là đa nền tảng |  |

### 3. Viết code lập trình theo hướng đối tượng

Vào **File > New > Application**

Xuất hiện cửa sổ làm việc gồm các thành phần sau

|  |  |
| --- | --- |
| - Đây là **Cửa sổ điều khiển chính** | |
| - Trong **Cửa sổ điều khiển chính** có **Menu chính** | |
| - Ở khoàng giữa là thành phần **Form** để thiết kế cửa sổ cho ứng dụng, sẵn sàng cho việc đặt các đối tượng lên trên |  |

- Bên trái là cửa sổ **Kiểm tra đối tượng** **Object Inspector**, giúp thiết lập các **Thuộc tính** và các **Sự kiện** cho đối tượng đặt lên Form

|  |  |
| --- | --- |
| - Các **Thuộc tính** ( **Properties** ) chứa dữ liệu ( **data** ) của đối tượng  **Ví dụ**: Đối tượng nút nhấn ( **Button** ) chứa các dữ liệu như   * **Kích thước** nút nhấn   **Tọa độ vị trí** nút nhấn trên Form   * **Tên** nút nhấn có thể là văn bản hay hình ảnh, tên nút được tạo bằng kiểu Font nào, màu sắc gì…   - Các **Sự kiện** ( **Events** ) để thiết lập cách tác động lên đối tượng ( **Phương thức** ) như click chuột, kéo thả đối tượng, di chuyển chuột qua …  - **Viết Code** là công đoạn thực hiện sau khi thiết lập các sự kiện  \* **Diễn giải**: Có thể thấy cấu trúc lập trình hướng đối tượng trong Trình biên dịch Lazarus là **Đối tượng** + **Data** + **Phương thức** |  |

- Cửa sổ lớn nhất là **Trình biên tập nguồn Source Editor** để viết code

- Dưới cùng là Cửa sổ thông báo **Message** để thông báo trong quá trình biên dịch

## PHẦN 5 - CHƯƠNG TRÌNH LAZARUS ĐẦU TIÊN

- Đây là vùng dành cho người mới bắt đầu

|  |  |  |
| --- | --- | --- |
| - Click nút |  | **t**rong trang **Hướng dẫn Lazarus**, tại mục 1. **Bắt đầu**, click chọn  **Chương trình đầu tiên của bạn** vàThực hiện các thao tác như trong hướng dẫn |

## PHẦN 6 - CÁC THƯ VIỆN TRONG FREE PASCAL & LAZARUS

- **Thư viện** (**Unit**) chứa các thủ tục, hàm, hằng số … được định nghĩa trước

- Thư viện trong Lazarus rất phong phú, được cộng đồng đông đảo những người lập trình **Lazarus** bổ sung, nó cung cấp nhiều chức năng sử dụng cho nhiều loại ứng dụng khác nhau.

- Trong số các thư viện thì có Thư viện **SysUtils** thường được sử dụng nhiều vì chứa việc mã cơ bản nhất cần thiết cho việc lập trình

- Để biết chi tiết về các Thủ tục, Hàm trong thư viện **SysUtils** được xây dựng như thế nào hãy vào địa chỉ websites

**http://www.math.uni-leipzig.de/pool/tuts/FreePascal/units/node21.html**

## PHẦN 7 - THƯ VIỆN NGƯỜI DÙNG TỰ TẠO & CÁCH TẠO

- Trong **Turbo Pascal** đều **Free Pascal** cũng có chức năng xây dựng Thư viện

- Trong Lazarus, cách tạo một thư viện mới là vào **File > New Unit** sẽ tự phát sinh mã có cấu trúc như sau

**unit Unit1;**

**{$mode objfpc}{$H+}**

**interface**

**uses Classes, SysUtils;**

**implementation**

**end.**

## PHẦN 8 - MINH HỌA CÁCH TẠO MỘT THƯ VIÊN

- Vào **File > New Unit**

- Viết code

- Lưu lại thành một tập tin ví dụ **MyUnit .pas**, thì tập tin **MyUnit.pas** trở thành một thư viện

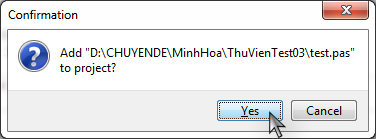
- Ứng dụng nào muốn sử dụng thư viện **MyUnit** cần phải khai báo **Uses MyUnit;**

- Sau khi khai báo, một Ứng dụng có thể gọi các **Thủ tục** (**Procedure**), **Hàm** ( **Function** ) của thư viện đó

Trong **Trình biên dịch Lazarus**, Các ứng dụng có thể truy cập vào các thư viện tự tạo với điều kiện:

1. Tập tin thư viện tồn tại trong cùng một thư mục với ứng dụng

2. Thêm thư viện vào một **Project**, bằng cách mở Thư viện Unit trong Lazarus sau đó chọn **Project > Add Editor File**



## PHẦN 9 - THỦ THUẬT KHAI THÁC MÃ NGUỒN MỞ TRONG LAZARUS

### 1. Danh sách các đối tượng thành phần

|  |  |
| --- | --- |
| - Nhấn tổ hợp phím **CTRL + ALT + P** sẽ hiện ra danh sách các đối tượng thường dùng, giúp chèn nhanh đối tượng trong danh sách vào Form bằng cách click vào đối tượng sau đó click lên Form |  |

### 2.Tra cứu mã nguồn mở của các thư viện, thủ tục, hàm … của Trình biên dịch Lazarus

|  |  |
| --- | --- |
| - Thủ thuật này giúp đi đến nơi mà đối tượng tra cứu ( hàm, thủ tục, đối tượng, ...) được khai báo trong một thư viện nào đó.  - Nhấn giữ phím **CTRL** rồi click chuột vào tên đối tượng, hàm hoặc thủ tục .  - Thủ thuật này rất hữu ích cho các bạn kiểm tra xem đối tượng đó đã được khai báo, liên kết hay chưa. | [http://1.bp.blogspot.com/-GuMbcKQrHqI/Ve21hQ8dreI/AAAAAAAACbE/p7XlYHB8VEE/s400/Untitled.png](http://1.bp.blogspot.com/-GuMbcKQrHqI/Ve21hQ8dreI/AAAAAAAACbE/p7XlYHB8VEE/s1600/Untitled.png) |

### 3. Danh sách thủ tục – Procedure list

|  |  |
| --- | --- |
| - Thủ tục là chương trình con (**Procedure**) trong ngôn ngữ Pascal.  - Nhấn **ALT + G** sẽ hiện ra danh sách các thủ tục có trong thư viện hiện tại.  - Đây là thủ thuật rất hữu ích khi phải viết một thư viện lên đến vài trăm thủ tục.  - Khi click vào tên thủ tục thì thủ tục đó được mở ngay mà không phải sử dụng phím mũi tên lên xuống để đi đến một thủ tục đang cần | [http://2.bp.blogspot.com/-WbjpPb2sFFY/Ve23Ia0k7jI/AAAAAAAACbQ/D-TrTPhZt0Q/s400/Untitled.png](http://2.bp.blogspot.com/-WbjpPb2sFFY/Ve23Ia0k7jI/AAAAAAAACbQ/D-TrTPhZt0Q/s1600/Untitled.png) |

### 4. Trình quản lý thứ tự của các đối tượng

|  |  |
| --- | --- |
| - Vào **View / Tab Order**  - Hộp thoại sẽ hiện ra thứ tự của - Có thể thay đổi thứ tự bằng cách click vào các phím mũi tên trên hộp thoại | <http://4.bp.blogspot.com/-wKdIJum8v2k/Ve24HxKObEI/AAAAAAAACbY/cOt4kp0TGQ8/s1600/Untitled.png> |

### 5. Làm sạch các mục

|  |  |
| --- | --- |
| - Vào **File / Clean Directory**  - Tính năng này rất hữu ích khi muốn dọn dẹp những tập tin tạm trong quá trình biên dịch . Ngoài ra, cũng có thể chỉ định một định dạng tập tin bất kỳ muốn loại bỏ. | [http://2.bp.blogspot.com/-gn7-ZCIsI64/Ve2_IJcVfoI/AAAAAAAACb8/srf3Po-sC_c/s400/Untitled.png](http://2.bp.blogspot.com/-gn7-ZCIsI64/Ve2_IJcVfoI/AAAAAAAACb8/srf3Po-sC_c/s1600/Untitled.png) |

### 6. Xuất mã nguồn dưới dạng HTML

|  |  |
| --- | --- |
| Vào **File / Export As Html**.  - Giúp thuận lợi trong việc chia sẻ code mà giữ nguyên mẫu màu sắc, định dạng các câu lệnh , Font … nên rất thân thiện | [http://1.bp.blogspot.com/-S3WX3mj8zHs/Ve2_44UcCZI/AAAAAAAACcE/oSVIaUe0BF4/s320/Untitled.png](http://1.bp.blogspot.com/-S3WX3mj8zHs/Ve2_44UcCZI/AAAAAAAACcE/oSVIaUe0BF4/s1600/Untitled.png) |

### 7. Những mẫu chương trình ví dụ

- Vào **Tools / Code Templates...,**

- Có thể chèn thêm vào những code có sẵn , chương trình sẽ hiện ra code gợi ý khi nhấn **Ctrl + J.**

## PHẦN 10 - LẬP TRÌNH VỚI ĐỐI TƯỢNG FORM

### 1. Công dụng

|  |  |
| --- | --- |
| - Đối tượng **Form** giúp tạo ra cửa sổ ứng dụng  - Từ Form, có thể sắp đặt, tổ chức các đối tượng lên đó tạo thành giao diện cho ứng dụng |  |

### 2. Tạo ứng dụng Lazarus mới

- Vào **File** > **New …** > **Applcation**, Đối tượng **Form** trống xuất hiện, sẵn sàng để đặt lên các đối tượng

|  |  |
| --- | --- |
| - **Biểu tượng** ( **Icon** ) của form mới **mặc định là biểu tượng của Lazarus**  - **Tiêu đề** ( **Caption** ) của form mặc định là **Form1** |  |

### 3. Lưu ứng dụng mới

|  |  |  |
| --- | --- | --- |
| - Tạo thư mục, ví dụ **App**  - Vào **File > Save As**, lưu vào thư mục vừa tạo  - Sửa tên file **project1.lpi ( lpi** là viết tắt của **Lazarus Project Information** ) thành **App.lbi** ( Đây là tập tin chính )  - Click nút **Save** | |  |
|  |  | |
| - Lưu tập tin **Unit1**, click nút **Save** |  | |
| - Sau khi lưu, cấu trúc file & folder trong thư mục **App** như sau |  | |

### 4. Một số thủ thuật cần chú ý :

|  |  |
| --- | --- |
| - **FORM** thường hay bị che bởi cửa sổ Trình biên tập nguồn **Source Editor** . Muốn Form hiện ra, có thể thực hiện :  **C1** : Nhấn phím **F12**  **C2** : Nhấn phím **SHIFT + F12** , trong hộp thoại **View Project Forms** , click chọn một tên **FORM**  **C3** : Click **FORM** button  - Trường hợp cửa sổ Trình biên tập nguồn **Source Editor** bị đóng có thể mở lại bằng cách:  **C1**: Vào **Windows**, click đánh dấu **Source Editor**  **C2**: Nhấn **CTRL+F12**, chọn nguồn trong danh sách  **C3**: Click nút Unit |  |

## PHẦN 11 - CÁC CÁCH THAY ĐỔI THUỘC TÍNH CỦA FORM

### 1. Đổi màu FORM

|  |  |
| --- | --- |
| - Click lên Form, có thể thay đổi thuộc tính bằng những cách dưới đây  **C1**: Trong thẻ **Properties**, chọn **Color** , click dấu ba chấm bên cạnh thẻ  Chọn hằng số màu    **C2**: Viết code **Form1.Color:=< Giá trị hằng màu >;**  - Xem bảng giá trị các hằng màu  **VD** : **Form1.Color:=clYellow;** |  |

### 2. Thay đổi biểu tượng ( Icon ) cho Form

#### a. Tải hình ảnh làm biểu tượng về máy

|  |  |
| --- | --- |
| - Tải về hình phù hợp chọn làm biểu tượng  - Lưu ảnh vào thư mục **App** |  |

#### b. Chuyển đổi hình ảnh thành tập tin Icon

|  |  |  |
| --- | --- | --- |
| - Mở websites **icoconvert.com**  - Thực hiện  **Bước 1** :  Click nút **Upload** để tải tập tin hình ảnh vừa tải về lên trang **Icoconvert.com**  **Bước 2** : Có thể giữ nguyên nếu không cắt xén hình ảnh  **Bước 3** : Chọn một kiểu dáng biểu tượng | |  |
| **Bước 4** : Định kích thước Icon **128x128** |  | |

|  |  |
| --- | --- |
| **Bước 5** :  - Click nút **Convert ICO**  - Click dòng **Download your icon(s)** |  |

|  |  |  |
| --- | --- | --- |
| - Lưu Icon file vào thư mục **App**  và lấy cùng phần tên với tên ứng dụng ta được **App.ico mới,** thiết lập thuộc tính cho **App.ico** là chỉ đọc ( **Read-only** )  - Click nút **Save** và **Yes**, lưu chồng lên **App.ico** cũ ( là **Tập tin ICO** mặc định của Lazarus cung cấp )  - Vào thẻ **Thuộc tính** **( Properties** ), chọn **Icon**, click **nút ba chấm** | |  |
| - Click nút **Load**. Sau khi nạp xong icon file, click **OK** |  | |

|  |  |
| --- | --- |
| - Vào **File > Quit**, chạy lại **App.lpi** và biên dịch lại mới có hiệu lực  - Tạo biểu tượng ứng dụng ra Desktop, click phải App file ( **App.exe** ) chọn **Send To > Desktop** |  |

- **Double click** biểu tượng để chạy ứng dụng, kết quả bước đầu thu được cửa sổ có biểu tượng mới vừa tạo, tuy nhiên tiêu đề của cửa sổ phải được đặt tên là tên của ứng dụng

### 3. Viết Code cho Form

|  |  |
| --- | --- |
| **C1**: **Double click** lên **Form**, sẽ tự động phát sinh thủ tục dành cho đối tượng **TForm1** với sự **kiện FormCreate**  **C2**: Chọn thẻ **Events** ( **Các sự kiện** ), chọn Sự kiện **OnActive** là **FormCreate**  **procedure TForm1.FormCreate(Sender: TObject);**  begin  end;  - Sự kiện **FormCreate** thường dùng khởi tao **Biến** hay **Hằng** |  |

### 4. Việc biên dịch

|  |  |
| --- | --- |
| - Để biên dịch vào **Run > Compile** hay **CTRL+F9** ( Có điểm khác với Free Pascal là **ALT+F9** ),  - **Message Panel** Sẽ xuất hiện khi có lệnh **Compile** ở đáy sẽ thông báo các vấn đề khi biên dịch giống với **Free Pascal** |  |

## PHẦN 12 PHƯƠNG PHÁP CHUNG THAY ĐỒI THUỘC TÍNH ĐỐI TƯỢNG

- Nhìn chung thường có 3 phương pháp thay đổi thuộc tính cho đối tượng

- Lấy ví dụ là đối tượng **Form** và thuộc tính **Tiêu đề** **Caption**

### 1. Thay đổi trực tiếp trong thẻ Thuộc tính Properties

|  |  |
| --- | --- |
| - Vào thẻ thuộc tính ( **Properties** ) , chọn **Caption**, sửa tên mặc định **Form1** thành tên tiêu đề mới là **My Application**    - Kết quả thu được như hình bên |  |

### 2. Thay đổi thuộc tính Form bằng cách viết code

**VD: Form1.Caption:=’ My Application ‘;**

### 3. Thay đổi thuộc tính của Form bằng cách di chuyển Form

- Khi Form đặt ở đâu khi chạy sẽ xuất hiện tại đúng vị trí đó

- Khi kéo Form di chuyển thì thuộc tính **Left, Top** bị thay đổi, đó là tọa độ góc trên trái của cửa sổ hoặc thuộc tính **Height** và **Width** thay đổi khi thay đổi kích thước cửa sổ Form

## PHẦN 13 - LẬP TRÌNH VỚI ĐỐI TƯỢNG NÚT NHẤN - BUTTON

|  |  |
| --- | --- |
| - Là đối tượng phổ biên nhất trong chương trình giúp người sử dụng tương tác với máy | <http://1.bp.blogspot.com/-DpDohEXY0CU/VXE8Jaj7AAI/AAAAAAAACAY/65eFA40bLzI/s1600/tbutton.png> |

**Sử dụng đối tượng lớp Tbutton**

- Mở thẻ **Standard**, click chọn Button sau đó click lên Form, đối tượng Button sẽ được chèn lên form

- Mặc định nút nhấn Button gắn với sự kiện **OnClick** với tên hành động là **Button1Click**

### Vô hiệu hóa Button

**Button1.Enabled := False;** Kết quả nút sẽ mờ đi

### Hiệu lực hóa Button

B**utton1.Enabled:=TRUE;**

### Thuộc tính canh chỉnh Button

- Để đặt **Button** ở đáy cửa sổ, click chọn **Button**

- Vào thuộc tính **Align** , chọn **alBottom**

- Tương tự có thể đặt Button ở đỉnh cửa sổ , chọn **alTop**

### Đối tượng SENDER

- Khi tương tác với một đối tương, **Sender** sẽ lưu trử tên trong thuộc tính **Name** của đối tượng đó kết quả **Sender=Name**

- Đối tượng **Sender** thuộc kiểu đối tượng **TObject**

**Thực hiện yêu cầu**

|  |  |
| --- | --- |
| - Khi click vào một đối tượng **Button** thì thuộc tính Tiêu đề **Caption** của **Form** sẽ hiển thị và thay đổi thành Tiêu đề **Caption** của **Button** đó |  |

**Viết code**

|  |  |
| --- | --- |
| **B1**: DoubleClick **Button1** sẽ phát sinh thủ tục  **procedure TForm1.Button1Click(Sender: TObject);**  **B2**: - Xóa kí tự 1 trên dòng tên thủ tục  \* **Lưu ý**: Phải sửa cả dòng khai báo giống như vây.  **procedure TForm1.ButtonClick(Sender: TObject);** |  |

**B3**: Thiết lập cả 3 nút có chung sự kiện là **ButtonClick**

**-Thực hiện**: Click vào **từng Button**, Vào **thẻ Events**, chọn sự kiện **Onclick** cùng là **ButtonClick**

**B4**: Gõ code cho sự kiện **ButtonClick** và chú ý cách viết code cho đối tượng **Sender**

\* **Lưu ý** : Thuộc tính **Caption** không ghi rõ chủ thể của nò thì đương nhiên thuộc sở hữu của **Form1** hay **From1.Caption**

**procedure TForm1.ButtonClick(Sender: TObject);**

**begin**

**if Sender=Button1 then Caption:=Button1.Caption;**

**if Sender=Button2 then Caption:=Button2.Caption;**

**if Sender=Button3 then Caption:=Button3.Caption;**

**end;**

- Nhấn **F9** để chạy chương trình và xem kết quả

- Cách viết như trên chưa gọn, có thể cải tiến như sau:

🡪 Thay **3 dòng If** trên bằng một dòng code 🡪 **Caption:=(Sender As TButton).Caption**

- Khi Button được nhấn thì **(Sender As TButton)** trả về thuộc tính **Name** của nút nhấn Button.

- Nhấn **F9** thấy rằng kết quả không đổi

- Để tránh **Sender** của **Button** bị sai lạc với **Sender** của đối tượng khác trên Form khi đó bổ sung câu lệnh If

**If Sender is TButton then** **Caption:=(Sender As TButton).Caption**

**Chạy Appication**

|  |  |
| --- | --- |
| - Để ý khi click **Button1** thì thuộc tính Tiêu để **Caption** của **Form1** bị thay đổi thành **Button1** , tương tự như vậy khi click các Button khác |  |

## PHẦN 14 - LẬP TRÌNH TẠO ỨNG DỤNG CÓ THÊM MỘT FORM THỨ HAI

- Trong thực tế có thể cần nhiều **Form**.

|  |  |
| --- | --- |
| **Các bước thực hiện :**  1. Tạo một ứng dụng mới và lưu **project** nó vào một thư mục mới có tên là **secondform** dưới dạng **secondform.lp**i.  2. Lưu **Unit** mới thành **main.pas** và Đặt thuộc tính **Name** cho Form là **fmMain**.  3. Thêm một Form thứ hai bằng cách click vào **File > New Form**. Lưu **Unit** mới này dưới dạng **second.pas** và ta đổi thuộc tính **Name** thành **fmSecond**.  4. Đổi tên thuộc tính tiêu đề **Caption** thành **'Second Form'**.  5. Trở về, click **CTRL + F12** chọn **main.pas** |  |

Click **OK** để đi đến **Main form** và đặt một button lên nó

|  |  |
| --- | --- |
| 6. Thêm vào sau dòng **implementation** : **uses second;** | **implementation**  **{$R \*.lfm}**  **uses second;** |

|  |  |
| --- | --- |
| 7. Trong sự kiện **OnClick event** của button, viết dòng code này: **fmSecond.Show;** | **procedure TfmMain.Button1Click(Sender: TObject);**  **begin**  **fmSecond.Show**  **end;** |

Chạy ứng dụng và click lên Button sẽ hiển thị form thứ hai

## PHẦN 15 - LẬP TRÌNH ĐỐI TƯỢNG NHÃN - LABEL

**Sử dụng đối tượng lớp TLabel**

|  |  |
| --- | --- |
| - Thường để giúp ghi chú lên Form  - Để tạo **Label** **,** Click **thẻ Standard**, đặt thành phần **TLabel** lên Form  - Để truy cập vào **Caption** của **Label1**, viết code **Label1.Cation** là giá trị kiểu **String** | **<http://4.bp.blogspot.com/-_wVSY1TzRGQ/VXE7wqONZ2I/AAAAAAAACAI/PWvjb3GfXys/s1600/tlabel.png>** |

|  |  |
| --- | --- |
| Ví dụ để hiện câu chào ta viết **Label1.Caption:='Chào các bạn';** |  |

|  |  |
| --- | --- |
| - **Label** chỉ nhận giá trị **kiểu Chuỗi**, những kiểu dữ liệu khác phải đổi sang chuỗi  - Cách đổi | **Labelx.Caption:= FloatToString (số);**  **Labelx.Caption:=IntToStr(số);**  **Labelx.Caption:=DateToStr(Date);**  **Labelx.Caption:=FormatDateTime(‘dd/mm/yyyy h:m:ss’,Now);**  **Labelx.Caption:=TimeToStr(Time);** |

## PHẦN 16 - LẬP TRÌNH VỚI ĐỐI TƯỢNG HỘP VĂN BẢN - EDIT BOX

|  |  |
| --- | --- |
| **Sử dụng đối tượng lớp TEdit trong thẻ Standard** | <http://2.bp.blogspot.com/-oJw4tR7k5WU/VXE8dFqg8aI/AAAAAAAACAw/8_cVdj8c_Us/s1600/tedit.png> |

|  |  |
| --- | --- |
| - Nhập văn bản vào đối tượng **EDIT**  - Cho phép thay đổi giá trị văn bản trong đối tượng **Edit**  - Đối tượng **Edit** thường được dùng đề nhập giá trị cho biến  - **EditX.Text**, **EditX.Caption** có tác dụng như tên biến, có điểm khác là chúng không phải khai báo | **EditX.Text:=<String>;**  **EditX.Caption:=<String>;** |

|  |  |
| --- | --- |
| **VD**: **Edit1.Caption:=’Hello’;**  **Edit2.Text:=’Chao’;** |  |

|  |  |
| --- | --- |
| **Đổi giá trị trong Edit Box sang số nguyên** | - Dùng hàm **StrToInt(Edit1.Text)** |
| **Đổi giá trị trong Edit Box sang số thực** | - Dùng hàm **StrToFloat(Edit1.Text)** |
| **Đổi số nguyên sang Text trước khi nạp vào Edit Box** | **Edit1.Text:=IntToStr(Number); (** Var Number:Integer; ) |
| **Đổi số thực sang Text trước khi nạp vào TextBox** | **Edit1.Text:=FloatToStr(Number); (** Var Number:Real; ) |
| **Định dạng kiểu ngày trong Edit Box** | **Edit1.Text:=FormatDateTime(‘dd/mm/yyyy’,Now)** |
| **Định dạng kiểu giờ trong Edit Box** | **Edit1.Text:=FormatDateTime(‘h:m:ss’,Now)** |
| **Định dạng cả ngày và giờ trong Edit Box** | **Edit1.Text:=FormatDateTime(‘dd/mm/yyyy h:m:ss’,Now** |

## PHẦN 17 - LẬP TRÌNH VỚI ĐỐI TƯỢNG HỘP THOẠI - DIALOG

- Hộp thoại là dạng cửa sổ Popup hiển thị một số thông tin và yêu cầu phản hồi của người dùng bằng cách:

🡪 Click nút nhấn sau khi đọc xong

🡪 Nhập văn bản hoặc cả hai.

- Lập trình viên có ít quyền kiểm soát định dạng, kích thước hoặc vị trí của các hộp popup này, nhưng có thể ảnh hưởng đến nội dung văn bản của chúng

### 1. Thủ tục tạo hộp thoại đơn giản - Message Dialogs

**procedure ShowMessage (const Msg: string);**

Thủ tục này thuộc thư viện **Dialogs**

Đây là hộp thoại thông báo đơn giản nhất, người sử dụng chỉ có một lưa chọn là nhấn OK



#### MINH HỌA: Tạo hộp thông báo có hai dòng thông báo

|  |  |
| --- | --- |
| **C1**: **Dùng điều khiển ngắt dòng sLineBreak** |  |
| **C2: Dùng mả kí tự #13#10**  **ShowMessage(‘ Text 1 ‘+#13#10+’ Text 2 ‘);**  **C3: Dùng điều khiển Kết thúc dòng** **LineEnding**  **ShowMessage(‘ Text 1 ‘+LineEnding+’ Text 2 ‘);** |  |

### 2. Tạo hộp thoại có thể tùy biến - Hàm MessageBox

**function Application.MessageBox(Text, Caption: PChar; Flags: LongInt): Integer;**

hay

**function Application.MessageBox(ThongBao, TieuDe: PChar; HangSo: LongInt): Integer;**

- **HangSo** để chỉ định các kiểu nút trong hộp thoại

- **HangSo** viết theo cú pháp **Hằng số biểu tượng + Hằng số kiếu nút nhấn + Hằng số nút mặc định**

\* Lưu ý hàm này có thể dùng như một thủ tục

|  |  |  |
| --- | --- | --- |
| **HẰNG SỐ KIỂU BIỂU TƯỢNG** | | |
| **MB\_ICONQUESTION** = | |  |
| **MB\_ICONWARNING**  = **MB\_ICONEXCLAMATION** | |  |
| **MB\_ICONERROR** = **MB\_ICONHAND** | |  |
| **MB\_ICONINFORMATION** | |  |
| **HẰNG SỐ KIỂU NÚT NHẤN** | | |
| **MB\_OK** | Hộp thoại chỉ có nút **OK** | |
| **MB\_OKCANCEL** | Hộp thoại có 2 nút **OK** và **CANCEL** | |
| **MB\_ABORTRETRYIGNORE** | Hộp thoại có 3 nút **ABORT RETRY** và **IGNORE** | |
| **MB\_YESNOCANCEL** | Hộp thoại có 3 nút **YES NO** và **CANCEL** | |
| **MB\_YESNO** | Hộp thoại có 2 nút **YES** và **NO** | |
| **MB\_RETRYCANCEL** | Hộp thoại có 2 nút **RETRY** và **CANCEL** | |

- Hàm **Application.MessageBox** trà về các giá trị kiểu Integer

|  |  |
| --- | --- |
| **IDOK** = **ID\_OK** = 1;  **IDCANCEL** = **ID\_CANCEL** = 2;  **IDABORT**  = **ID\_ABORT** = 3;  **IDRETRY** = **ID\_RETRY** = 4;  **IDIGNORE** = **ID\_IGNORE** = 5;  **IDYES** = **ID\_YES** = 6;  **IDNO** = **ID\_NO** = 7;  **IDCLOSE** = **ID\_CLOSE** = 8;  **IDHELP** = **ID\_HELP** = 9; |  |

Hằng số nguyên trả về từ **MessageBox** cho biết nút nào được nhấn, định nghĩa nút đầu tiên, thứ hai hoặc thứ ba là mặc định

MB\_DEFBUTTON1 = $00000000;

MB\_DEFBUTTON2 = $00000100;

MB\_DEFBUTTON3 = $00000200;

MB\_DEFBUTTON4 = $00000300;

|  |  |
| --- | --- |
| Nó có thể được gọi như một lời gọi thủ tục |  |

Khai báo thư viện

uses Forms, **Dialogs**, **LCLType**;

**procedure DisplayMessageBox;**

var Reply, BoxStyle: Integer;

begin

BoxStyle := **MB\_ICONQUESTION + MB\_YESNO;**

// Dùng như một hàm

Reply := **Application.MessageBox('Press either button', 'MessageBoxDemo', BoxStyle);**

// Dùng như một thủ tục

if Reply = **IDYES** then **Application.MessageBox('Yes ', 'Reply',MB\_ICONINFORMATION)**

else **Application.MessageBox('No ', 'Reply', MB\_ICONHAND);**

end;

|  |  |
| --- | --- |
| Nếu click **YES** , sẽ xuất hiện **MessageBox** như sau : |  |

|  |  |
| --- | --- |
| Nếu click **NO**, **MessageBox** sẽ xuất hiện như sau: |  |

**Thủ thuật**: Lưu ý rằng trong ví dụ này, các chuỗi ‘**Yes ‘** và ‘**No ‘** đã được đệm bằng dấu cách, nếu không hộp sẽ không đủ rộng để hiển thị chú thích đúng cách

**procedure DisplayMessageBox2;**

var

Reply, BoxStyle: Integer;

begin

BoxStyle := **MB\_ICONQUESTION + MB\_ABORTRETRYIGNORE**; // tạo 3 nút **Abort, Retry, Tgnore**

Reply := **Application.MessageBox('Hãy click nút để chọn', 'MessageBoxDemo', BoxStyle);**

if Reply = **IDABORT** then **Application.MessageBox('Khởi tạo lại ', 'Abort',MB\_ICONINFORMATION)**

else if Reply = **IDRETRY** then **Application.MessageBox('Hãy thử lại ', 'Retry', MB\_ICONHAND)**

else **Application.MessageBox('Hãy lờ đi ', 'Ignore', MB\_ICONHAND)**

end;

### 3. Hàm MessageDLG – Tạo hộp thông báo phức tạp

**function MessageDlg(const aMsg: string; DlgType: TMsgDlgType;**

**AButtons: TMsgDlgButtons; HelpCtx: Longint): Integer;**

**function MessageDlg(const aCaption, aMsg: string; DlgType: TMsgDlgType;**

**AButtons: TMsgDlgButtons; HelpCtx: Longint): Integer;**

- Hai phiên bản của hàm này, tức là tham số '**Caption**' (**Tiêu đề**) đầu tiên là tùy chọn; nếu bỏ qua, chú thích bị thiếu

- Đây là hộp thoại thông điệp đầy đủ và phức tạp nhất và cho phép lập trình viên kiểm soát đáng kể sự xuất hiện của hộp thoại. Các tham số xác định loại hộp và biểu tượng của nó là các loại thay vì hằng số nguyên, và các nút có thể được chỉ định như một tập hợp trong các dấu ngoặc vuông, ví dụ **[mbRetry, mbIgnore, mbAbort, mbCancel]**.

- Tham số **HelpCtx** hiện không được triển khai và phải được đặt thành 0. Giá trị trả về từ hàm là danh tính của nút được nhấn, được biểu thị dưới dạng số nguyên (xem định nghĩa bên dưới, [mrNone..mrAll])

**Ví dụ**:

Uses Forms, Dialogs, **LCLType**, Controls;

**procedure TryMessageDlg;**

begin

if MessageDlg('Question', 'Do you wish to Execute?', mtConfirmation, [mbYes, mbNo, mbIgnore],0) = mrYes

then { Execute rest of Program };

end;

Nếu click **Yes** để thực hiện , hãy đưa khối lệnh váo vị trí { Execute rest of Program }

### 4. Hộp thoại QuestionDlg

Trong ví dụ dưới đây cho phép thay đổi **tiêu đề đối tượng button** và thiết lập **đối tượng button mặc định**

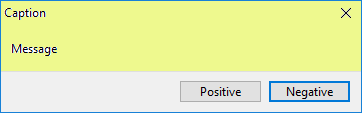
**case QuestionDlg ('Caption','Message',mtCustom,[mrYes,'Positive', mrNo, 'Negative', 'IsDefault'],'') of**

**mrYes: QuestionDlg ('Caption','So you mean „Yes“',mtCustom,[mrOK,'That is right'],'');**

**mrNo: QuestionDlg ('Caption','Oh, you mean „No“',mtCustom,[mrOK,'Exactly'],'');**

**mrCancel: QuestionDlg ('Caption','You canceled the dialog with ESC or close button.',mtCustom,[mrOK,'Exactly'],'');**

**end;**



|  |  |
| --- | --- |
| Nếu click **Positie button** |  |
| Nếu click **Negative button** |  |

**Lưu ý**: Nếu người dùng hủy bỏ hộp thoại bằng phím ESC, dù không có nút '**IsCancel**' được xác định, nhưng vẫn sẽ trả về **mrCancel**

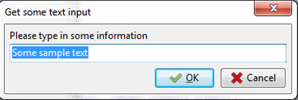
### 5. Hàm InputBox

**function InputBox (const ACaption, APrompt, ADefault: string): String;**

**Hộp nhập văn bản** : hiển thị một thông báo và chờ user nhập văn bản

**function InputBox(const ACaption, APrompt, ADefault: String): String;**

Hiển thị một hộp có tiêu đề và lời nhắc được xác định và dự kiến ​​người dùng nhập vào một hộp văn bản. Một chuỗi mặc định có thể tùy chọn được hiển thị trong hộp văn bản. Chuỗi do người dùng nhập hoặc mặc định được trả về dưới dạng kết quả hàm

**Ví dụ** :

- Phải khai báo thư viện **LCLType** và **Dialogs**

**uses**

**Forms, LCLType, Dialogs, Controls;**

**var**

**UserString: string;**

**begin**

**UserString := InputBox('Get some text input','Please type in some information', 'Some sample text');**

**ShowMessage(UserString)**

**end;**

### 6. Hàm InputQuery

**function InputQuery(const ACaption, APrompt : String; MaskInput : Boolean; var Value : String) : Boolean;**

**function InputQuery(const ACaption, APrompt : String; var Value : String) : Boolean;**

. Hai phiên bản của hàm này sẽ hiển thị dấu nhắc người dùng và dữ liệu văn bản;

- Tham số boolean **MaskInput** xác định dữ liệu nhập của người dùng có bị che dấu bởi dấu hoa thị trong hộp nhập văn bản ( giống như nhập mật khẩu ) hay không, trong khi hộp thoại thứ hai bỏ qua thuộc tính này.

- Bỏ qua tham số **MaskInput** tương đương với thiết lập nó FALSE.

#### MINH HỌA: - Tạo hộp thông báo nhập văn bản và che dấu văn bản được nhập bởi dấu \*

- Sau đó cho nhìn thấy văn bản đã nhập

uses

Forms, **LCLType**, Dialogs, Controls;

**procedure TryInputQuery;**

**var QueryResult: Boolean;**

**UserString: string;**

**begin**

**if InputQuery('Question', 'Type in some data', TRUE, UserString)**

**then ShowMessage(UserString)**

**else**

**begin**

**InputQuery('Don''t be silly', 'Please try again', UserString);**

**ShowMessage(UserString);**

**end**

**end;**

|  |  |
| --- | --- |
| Chạy chương trình, kết quả thu được |  |
| Click **OK** dòng văn bản bí mật sẽ được hiển thị |  |

### 7. Hàm PasswordBox

**Function PasswordBox(const ACaption, APrompt : String) : String;**

Hoạt động tương tự như hàm **InputQuery** với **MaskInput = TRUE;** sự khác biệt là mật khẩu đã nhập vào được trả về là kết quả của hàm (như **InputBox**).

Hằng số và Kiểu được sử dụng trong hộp thoại tin nhắn

Một số hằng số và loại có liên quan để sử dụng với các hộp thoại được xác định trước trong thư viện **LCLType**

## PHẦN 18 - MINH HỌA TẠO ỨNG DỤNG GAME ĐƠN GIẢN

**Ý tưởng phát triển**

|  |  |
| --- | --- |
| **Tạo một chương trình game mà chỉ sử dụng 4 đối tượng cơ bản nhất:**  - Form  - Label  - Edit  - Button  - Hộp thông báo |  |

**Thiết lập thuộc tính Name của các nút nhấn**

Nút **Check** 🡪 **btnCheck**

Nút **New Game** 🡪 **btnNewGame**

Nút **Exit**  🡪 **btnExit**

- Người chơi nhập số

- Nhấn nút **Check**

- Khi khởi động chương trình , biến nguyên **RandomNumbe**r được khởi tạo và là biến toàn cục , thực hiện chèn dòng khai báo biến **RandomNumber** sau dòng private

**TForm1 = class(TForm)**

**btnCheck: TButton;**

**btnNewGame: TButton;**

**btnExit: TButton;**

**Edit1: TEdit;**

**Label1: TLabel;**

**procedure FormCreate(Sender: TObject);**

**private**

**var RandomNumber:Integer;**

**public**

**end;**

- Kế đó nhờ sự kiện **FormCreate**, Giá trị ngẫu nhiên **RandomNumer** được kích hoạt và nạp giá trị **ngẫu nhiên từ 1 đến 100**

- Trước khi kích hoạt **RandomNumber** , phải khởi tạo thủ tục tạo bộ số ngẫu nhiên **Randommize;** , sau khởi tạo hàm **random()** mới có hiệu lực

- **Random(100);** trả về **giá trị ngẫu nhiên** trong phạm vi **từ 0 đến 99**. Nên **Random(100)+1** trả về **giá trị ngẫu nhiên** trong phạm vi từ 1 đến 100

**procedure TForm1.FormCreate(Sender: TObject);**

**begin**

**Randomize;**

**RandomNumber:=Random(100)+1**

**end;**

- Khi chạy CT người chơi phải đoán số và gõ giá trị số trong phạm vi từ 1 đến 100 vào thành phần **Edit Box**

- Giá trị số trong khung **Edit Box** là giá trị văn bản được lưu giữ trong biến **Edit1.Text**

- Phải dùng hàm **StrToInt()** để chuyển đổi số định dạng văn bản trong **Edit Box** sang số định dạng số và lưu vào biến **GuessNumber**

|  |  |
| --- | --- |
| - **Kiểm tra biến GuessNumber** :  🡪 Nếu giá trị nằm ngoài phạm vi 1 🡪 100 thì xuất thông báo yêu cầu nhập lại bằng thủ tục **ShowMessage()**  🡪 Nhỏ hơn **Random Number** thì xuất thông báo :  ‘ Hãy nhập số lớn hơn ‘  🡪 Lớn hơn **Random Number** thì xuất thông báo :  ‘ Hãy nhập số nhỏ hơn ‘  🡪 Lớn hơn **Random Number** thì xuất thông báo :  ‘ Bạn đã đoán đúng ‘ |  |

- Click kép lên nút **Check** chèn vào đoạn mã sau :

**procedure TForm1.btnCheckClick(Sender: TObject);**

**var GuessNumber:Integer;**

**begin**

**GuessNumber:=StrToInt(Edit1.Text);**

**If (GuessNumber<0) or (GuessNumber>100) then ShowMessage('Hãy nhập lại giá trị trong phạm vi từ 1 đến 100');**

**If GuessNumber<RandomNumber then ShowMessage(' Hãy nhập giá trị lớn hơn');**

**If GuessNumber>RandomNumber then ShowMessage(' Hãy nhập giá trị nhỏ hơn');**

**If GuessNumber=RandomNumber then ShowMessage(' Bạn đã đoán đúng');**

**end;**

- Sau khi người chơi đoán đúng , có thể click **nút** **New Game** để tiếp tục game hoặc click **nút Exit** để thoát khỏi trò chơi

- Sau khi click **New Game** , giá trị cũ trong **Edit Box** phải được xóa đi và khởi tạo lại giá trị số ngẫu nhiên

- Click 2 lần lên **nút New Game** , chèn đoạn mã sau :

**procedure TForm1.btnNewGameClick(Sender: TObject);**

**begin**

**Edit1.Text:='';**

**Randomize;**

**RandomNumber:=Random(100)+1**

**end;**

- Nếu người chơi muốn thoát khỏi chương trình , có thể click nút Exit

- Click 2 lần lên **nút Exit** , chèn đoạn mã sau :

**procedure TForm1.btnExitClick(Sender: TObject);**

**begin**

**Close()**

**end;**

## PHẦN 19 - LẬP TRÌNH VỚI ĐỐI TƯỢNG KHUNG VĂN BẢN MEMO

|  |  |
| --- | --- |
| **Sử dụng đối tượng lớp TMemo** ( Trong thẻ **Standard** )  - Cho phép tương tác với nhiều dòng văn bản | <http://3.bp.blogspot.com/-PFurGZEI9SI/VXE8X-e9jaI/AAAAAAAACAo/H8V7JhvlGIQ/s1600/tmemo.png> |

**Viết code**

|  |  |
| --- | --- |
| **Gán một dòng văn bản vào memo**  - Muốn gán nhiều dòng thì dùng nhiều dòng lệnh  - Nếu trước đó có văn bản thì thủ tục này sẽ chèn văn bản xuống dòng kế | **begin**  **Memo1.Lines.Clear; //delete all lines of Memo1**  **Memo1.Lines.Add('This is the first line.'); //add a line**  **Memo1.Lines.Add('This is the second line.');**  **Memo1.Lines.Add('This is the third line.');**  **Memo1.Lines.Add('etc.');**  **end;** |

|  |  |
| --- | --- |
| **Gán chỉ một dòng văn bản**  Nếu trước đó có chứa văn bản sẽ bị xóa hết & thay bằng văn bản của **MemoX.Text** | **MemoX.Text: = 'đây là văn bản đơn lẻ';** |

|  |  |
| --- | --- |
| **Dùng điều kiển LineEnding trong Memo1Text**  - Có tác dụng điều khiển xuống dòng  - Memo1.Text sẽ xóa văn bản trước đó trước khi chèn nội dung của Memo1.Text | **Memo1.Text: = ‘This’ + LineEnding + ‘là’ + LineEnding + ‘a’ + LineEnding + ‘multiline’ + LineEnding + ‘text’;** |

|  |  |
| --- | --- |
| **Gán một đối tượng thuộc lớp TstringList vào đối tượng Memo**  Kết quả thu được | **var**  **myStringList: TStringList;**  **begin**  **myStringList:=TStringList.Create; //Create my StringList**  **myStringList.Add('This is the first line.'); //add a line**  **myStringList.Add('This is the second line.');**  **myStringList.Add('This is the third line.');**  **myStringList.Add('etc.');**  **Memo1.Lines.Assign(myStringList); //assign text content**  **myStringList.Free; //free my StringList**  **end;** |

|  |  |
| --- | --- |
| **Đọc một dòng**  **Cách 1**:  - Nếu muốn biết nội dung của một dòng cụ thể, thực hiện như hình bên  - Lưu ý, chỉ mục của **TMemo.Lines** là **0**, nghĩa là dòng đầu tiên sẽ là: **myString: = Memo1.Lines [0];** | **myString: = Memo1.Lines [Index];**  **ShowMessage(Memo1.Lines[2]);** |

|  |  |  |
| --- | --- | --- |
| **Cách 2**:  - Dùng thủ tục | **Memo1.Lines.GetNameValue(Index:Integer;Out AName,AValue:String);**  **procedure TForm1.Button1Click(Sender: TObject);**  **var A\_Name,A\_Value:String;**  **begin**  **Memo1.Lines.GetNameValue(0,A\_Name,A\_Value);**  **ShowMessage(A\_Value)**  **end;** | |
| **Chọn văn bản**  - Có thể chọn văn bản trong Memo giống như trong các trình soạn thảo văn bản  - Sữ dụng các chức năng **Copy, Cut, Paste, Undo** … bằng cách sử dụng các tổ hợp phím bình thường để giao tiếp với các phần mềm khác trong Windows  - Cách hiển thị văn bản được chọn | | **procedure TForm1.Button1Click(Sender: TObject);**  **begin**  **ShowMessage(Memo1.SelText);**  **end;**  Chạy & xem kết quả  Sau khi bôi đen để chọn, click Button sẽ hiển thị văn bản được chọn |

|  |  |
| --- | --- |
| **Canh chỉnh văn bản trong Memo**  - Định nghĩa của kiểu **TAligment**  **Type TAlignment = (taLeftJustify, taRightJustify, taCenter)** | **procedure TForm1.Button1Click(Sender: TObject);**  **var A\_Lign:TAlignment;**  **begin**  **A\_Lign:=taCenter;**  **Memo1.Alignment:=A\_Lign;**  **end;** |
| **Xóa dòng tại vị trí chỉ định trong Mem**o  - Thực hiện xóa dòng vị trí 0 | **procedure TForm1.Button1Click(Sender: TObject);**  **begin**  **Memo1.Lines.Delete(0);**  **end;** |

## PHẦN 20 - MINH HỌA TẠO TRÌNH SOẠN THẢO VĂN BẢN ĐƠN GIẢN

|  |  |
| --- | --- |
| Vào Thẻ DiaLogs đặt các đối tượng sau lên Memo:  **FindDialog**  **SaveDialog**  **OpenDialog** |  |

**Bước 1**: **Khởi tạo Form**

|  |  |
| --- | --- |
| - Xóa chữ Memo1  - Double Click lên Form | **procedure TForm1.FormCreate(Sender: TObject);**  **begin**  **Memo1.Lines.Clear**  **end;** |

**Bước 2**: **Ghi một dòng văn bản vào trong Memo**

|  |  |
| --- | --- |
| - Có thể ghi thủ công ( hơi lâu )  - Dùng nút Ghi Text để ghi cho nhanh | **procedure TForm1.Button1Click(Sender: TObject);**  **begin**  **Memo1.Lines.Add('This is a Memo');**  **end;** |

**Bước 3**: **Lưu lại nội dung trong Memo**

|  |  |
| --- | --- |
| - Lưu lại theo phần mở rộng TXT | **procedure TForm1.Button2Click(Sender: TObject);**  **begin**  **if SaveDialog1.Execute then**  **Memo1.Lines.SaveToFile(SaveDialog1.FileName);**  **end;** |

**Bước 4: Mở file TXT trong Memo**

|  |  |
| --- | --- |
|  | **procedure TForm1.Button3Click(Sender: TObject);**  **begin**  **if OpenDialog1.Execute then**  **Memo1.Lines.LoadFromFile(OpenDialog1.FileName);**  **end;** |

**Bước 5**: **Tìm văn bản trong Memo**

**procedure TForm1.Button4Click(Sender: TObject);**

**var**

**k,l: integer;**

**SearchText:String;**

**begin**

**SearchText:=Edit1.Text; // Lấy giá trị Text cần tìm trong Edit**

**k := Pos( SearchText, Memo1.Lines.Text ); // Vị trí ban đầu nếu tìm thấy sẽ <>0**

**if k>0 then**

**begin**

**l:=length( SearchText ); // Chiều dài Text tìm thấy**

**ShowMessage('Vị trí tìm thấy bắt đấu là vị trí thứ '+IntToStr(k)+' đến vị trí thứ '+IntToStr(k+l-1));**

**end**

**else**

**ShowMessage(' Không tìm thấy '); //**

**end;**

### Chọn Font trong Memo

|  |  |
| --- | --- |
| Double click lên nút font, thực hiện | **procedure TForm1.Button1Click(Sender: TObject);**  **begin**  **if FontDialog1.Execute then memo1.font:=FontDialog1.font;**  **end;** |

**Chạy & xem kết quả**

|  |  |
| --- | --- |
|  |  |

### Hiển thị ngày giờ trong Memo

|  |  |
| --- | --- |
| **Bước 1**: Tạo Form như hình bên  **Bước 2** : Click lên Form để phát sinh code |  |

|  |  |
| --- | --- |
| Gõ dòng **Memo1.Lines.Clear** để xóa văn bản trong Memo1 | **procedure TForm1.FormCreate(Sender: TObject);**  **begin**  **Memo1.Lines.Clear**  **end;** |

**Bước 3**: Click lên **Date&Time** button để phát sinh code

|  |  |
| --- | --- |
| **procedure TForm1.Button1Click(Sender: TObject);**  **var Y,M,D,H,MN,S,MS:word;**  **begin**  **DecodeDate(Now,Y,M,D);**  **DecodeTime(Now,H,MN,S,MS);**  **Memo1.Lines.Add('Ngày '+IntToStr(D)+' Tháng '+IntToStr(MN)+' Năm '+IntToStr(Y));**  **Memo1.Lines.Add(IntToStr(H)+' Giờ '+IntToStr(MN)+' Phút '+IntToStr(S)+' Giây '+IntToStr(MS)+' Mili Giây ');**  **end;**  Chạy & xem kết quả |  |

## PHẦN 21 - MINH HỌA XỬ LÝ THÔNG TIN TRÊN CÁC THIẾT BỊ LƯU TRỮ

**Ý tướng phát triển:**

- Xây dựng ứng dụng có khả năng tổ chức thông tin trên các hệ thống lưu trữ

- Tổ chức & khai thác thông tin một cách hiệu quả

### A. LƯU THÔNG TIN

**Sử dụng đối tượng lớp TSaveDialog** ( trong Thẻ **Dialog**s )

|  |  |
| --- | --- |
| **Thưc hiện**:  - Tạo ứng dụng soạn thảo chương trình Pascal  - Có mode lọc tập tin có phần mở rộng .TXT và .PAS  - Ưu tiên cho tập tin có phần mở rộng. PAS  - Thiết lập thư mục ngầm định hay thư mục ưu tiên để lưu file  Cách thiết lập thuộc tính cho đối tượng **SaveDialog** cũng tương tự như **OpenDialog** |  |
| Kết quả tập tin **.PAS** được mặc định ưu tiên để **SAVE** |  |

**Viết CODE cho SAVE button**

|  |  |
| --- | --- |
| - Đặt đối tượng **SaveDialog** lên **Form**  - **Double click** nút **Save**, điền dòng code sau  - Kết quả trong Memo1 hiển thị đường dẫn đã Save tập tin | **If SaveDialog1.Execute then Memo1.Lines.SaveToFile(SaveDialog1.FileName);** |

### B. MỞ THÔNG TIN

**Sử dụng đối tượng lớp TOpenDialog** ( trong Thẻ **Dialog**s )

|  |  |
| --- | --- |
| - Click đối tượng **OpenDialog1**  - Trong thuộc tính **DefaultExt** gõ kiểu file mặc định là **.PAS**  - Vào thuôc tính **Filter** click nút … để mở hộp thoại khai báo bộ lọc , gõ **\*.TXT , \*.PAS**  - Quan sát khung thuộc tính Filter |  |

|  |  |
| --- | --- |
| - Vào thuộc tính **InitiaDir** khai báo đường dẫn thư mục ngầm định chứa file và khi mở sẽ ưu tiên mở thư mục ngầm định  - Kết quả khi Mở File sẽ vào trực tiếp vị trí **D:\Programming\Lazaru** |  |

**Viết CODE cho LOAD button**

|  |  |
| --- | --- |
| - Double click nút Load, chèn dòng code sau: | **If OpenDialog1.Execute then Memo1.Lines.LoadFromFile(OpenDialog1.FileName);** |
| - Khi click nút **OPEN FILE** , chọn file thì đường dẫn file xuất hiện trong khung Edit  **Nhấn phím F9, chạy thử ứng dụng**  - Tạo nội dung , click nút **Save**  - Xóa nội dung trong **Memo1**  - Click nút **Load** để nạp lại nội dung đã lưu  **Chạy Application** |  |

Click nút **OPEN FILE**

## PHẦN 22 - MINH HỌA THAY ĐỔI GIAO DIỆN ỨNG DỤNG

**Ý tưởng phát triển**:

- Giao diện không chỉ hình thức mỹ thuật, còn đòi hỏi tổ chức các chuyên mục khoa học sao cho thân thiện với người dùng

- Người lập trình phải làm chủ được giao diện cho sản phẩm của mình, giữ vai trò trung gian giữa người dùng và máy tính, giúp người dùng dễ dàng thực hiện công việc của họ hiệu quả

**Sử dụng đối tượng lớp TColorDialog**

**Thiết kế FORM**

**Viết Code cho CHOOSE COLOR BUTTON :**

|  |  |
| --- | --- |
| **procedure TForm1.Button1Click(Sender: TObject);**  begin  ColorDialog1.Execute;  Edit1.Font.Color:=ColorDialog1.Color;  end;  **procedure TForm1.Edit1Change(Sender: TObject);**  begin  Edit1.Text:=' Hello ';  ShowMessage(' Hãy click Choose Color Button để chọn màu ');  end; |  |

Thủ tục **Edit1Change** sẽ phát hiện sự thay đổi trong khung **Edit1** , nếu có sẽ làm xuất hiện câu chào ‘ HELLO ‘ và xuất thông báo ' Hãy click **Choose Color Button** để chọn màu '

**Run Application**

- Click nút **CHOOSE COLOR** , chọn màu trên bảng màu, ví dụ: Chọn Red

- Click **OK** , Văn bản trong **EDIT BOX** có màu đỏ

|  |  |
| --- | --- |
|  |  |

## PHẦN 23 - MINH HỌA TƯƠNG TÁC VỚI TRÌNH DUYỆT & TRANG WEB

**Ý tưởng phát triển**

- Phát triển ứng dụng web cải tiến công việc

- Dịch vụ web phát triển ngày càng tăng, ví dụ các cuộc thi, hội thi qua mạng ngày càng nhiều, HS thường phải gõ địa chỉ website nhất là các websites có địa chỉ dài, chưa kể gõ sai gây ra tiêu tốn nhiều thời gian

- Thu gọn công việc này với một nút nhấn như một ứng dụng bình thường

**Thực hiện**

|  |  |
| --- | --- |
| - Tạo Form & 1 nút nhấn  - Khi click vào nút nhấn 🡪 Mở trình duyệt với trang Home Page Lazaus |  |

**OpenURL là cách đơn giản nhất**

|  |  |
| --- | --- |
| Thư viện **LCLIntf** không là mặc định, nên phải khai báo trước khi sử dụng thủ tục **OpenUR**L | **uses LCLIntf;**  **...**  **OpenURL('http://www.lazarus.freepascal.org');** |

**Phương thức FindingDefaultBrowser ( Tìm trình duyệt web mặc định )**

- Thư viện **LCL LazhelpHTML** chứa một **Class THTMLBrowserHelpViewer** để khởi động trình duyệt web cho Hệ thống trợ giúp LCL.

- Có thể sử dụng phương thức **FindDefaultBrowser** để tìm trình duyệt mặc định và tham số khởi động.

**Thực hiện yêu cầu**

|  |  |
| --- | --- |
| Khi click **Button** hiển thị :  - Đường dẫn của Trình duyệt mặc định  - Tham số của trình duyệt mặc định |  |

|  |  |
| --- | --- |
| **Bước 1** : Thư viện **LCLProc**, **LazHelpHTML** không mặc định phát sinh nên phải khai báo | **uses**  ..., **LCLProc, LazHelpHTML**; |

|  |  |
| --- | --- |
| **Bước 2**: **Double click** lên **From** và viết code  Xóa trắng đối tượng Memo | **procedure TForm1.FormCreate(Sender: TObject);**  **begin**  **Memo1.Lines.Clear;**  **end;** |

|  |  |
| --- | --- |
| **Bước 3**: **Double click Button1** và viết code cho sự kiện **OnClick** là **Button1Click** | **procedure TForm1.Button1Click(Sender: TObject);**  **var v: THTMLBrowserHelpViewer;**  **BrowserPath, BrowserParams: String;**  **begin**  **v:=THTMLBrowserHelpViewer.Create(nil);**  **v.FindDefaultBrowser(BrowserPath,BrowserParams);**  **debugln(['Path=',BrowserPath,'Params=',BrowserParams]);**  **Memo1.Lines.Add('Path='+BrowserPath);**  **Memo1.Lines.Add('Params='+BrowserParams);**  **v.Free;**  **end;** |

## PHẦN 24 - LẬP TRÌNH MÔ PHỎNG THIẾT BỊ ĐÈN TÍN HIỆU GIAO THÔNG

**Ý tưởng phát triển**

- Ý tưởng tạo ra các giáo cụ trực quan cho công việc giảng dạy và học tập

- Làm quen với thực tiễn thông qua lập trình điều khiển thiết bị đèn tín hiệu giao thông

- Đặt trường hợp phải thay đổi thời gian bật tắt từng đèn phải làm sao để việc điều khiển giao thông có hiệu quả

**Các lớp đối tượng sử dụng**: **TPanel** , **TShap** , **TTime**r

|  |  |
| --- | --- |
| **1. TPanel**  - Trong thẻ **Standard** , Click **Tpanel** , click lên **Form**  - Canh chỉnh , tạo khung chữ nhật như trong hình  - Trong thuộc tính **Color** , chọn giá trị **clMenuText** tương ứng với màu đen    **2. TShap**  - Trong thẻ **Additional** , Click **TShap** , tạo 3 đối tượng **Shape1** , **Shape2** , **Shape3** tương ứng với 3 đèn tín hiệu giao thông  - Trong thuộc tính **Shap** , chọn giá trị **stCircle** tương ứng cho 3 đối tượng **TShap** trên , thuộc tính **Visible** chọn trị **False** ( không hiển thị các **Shape** )  - Click dấu > để mở nhánh thuộc tính **Brush** , chọn **Color** , lần lượt chọn giá trị :  🡪 **clRed** tương ứng với màu đỏ cho **TShap1**  🡪 **clYellow** tương ứng với màu vàng cho **TShap2**  🡪 **clLime** tương ứng màu xanh lá cho **TShap3** |  |

**3. Tạo đối tượng TTIMER**

- Trong thẻ **System** , click **TTimer** , click lên **Form**

**4. Viết Code cho các đối tượng**

|  |  |
| --- | --- |
| **Viết code cho đối tượng TTIMER**  - Double click lên đối tượng TTIMER  - Khai báo biến toàn cục | **var Form1: TForm1;**  **sec:Integer;** |

**Thuật toán :**

**Cách 1** :

- Double click đối tượng Timer để viết code

- Dùng thủ tục inc(số giây)

- Khi số giây = 12 trở lại giá trị 0

- Khi số giây có giá trị từ 1 🡪 5 ,

🡪 cho xuất hiện đối tượng **Shape1** **màu đỏ** 🡪 **Shape1.Visible:=True;**

🡪 làm biến mất đối tượng **Shape2 và 3** **Shape2.Visible:=False;**

**Shape3.Visible:=False;**

- Khi số giây = 7

🡪 Làm xuất hiện đối tượng **Shape 2 vàng** 🡪 **Shape1.Visible:=True;**

🡪 tương tư như trên các đối tượng còn lại biến mất

- Khi số giây có giá trị từ 8 🡪 11

🡪 Làm xuất hiện đối tượng **Shape 3 xanh** 🡪 **Shape3.Visible:=True;**

🡪 tương tư như trên các đối tượng còn lại biến mất

- Lặp lại các thao tác trên

**Cách 2** : Thuật toán :

- Ban đầu tất cả các đèn được hiển thị nên 🡪 **Shape1.Visible:=True;**

**Shape2.Visible:=True;**

**Shape3.Visible:=True;**

- Khi số giây = 12 , số giây trở về 0

- Khi số giây từ 0 🡪 5 :

🡪 **Shape1.Brush.Color:=ClRed;** // Shape1 bật sáng bằng cách thay bằng màu đỏ tươi

🡪 **Shape2.Brush.Color:=ClOlive**; // Shape2 bị mờ với màu vàng tối

🡪 **Shape2.Brush.Color:=ClGreen**; // Shape3 bị mờ với màu xanh lục tối

- Khi số giây =7

🡪 **Shape2.Brush.Color:=ClYellow; //** Shape2 bật sáng bằng cách thay bằng màu vàng tươi

🡪 **Shape1.Brush.Color:=ClMaroon; //** Shape1 bị mờ với màu đỏ tối

🡪 **Shape3.Brush.Color:=ClGreen; //** Shape3 bị mờ với màu xanh lục tối

- Khi số giây từ 8 🡪 11

🡪 **Shape3.Brush.Color:=ClLime; //** Shape3 bật sáng bằng cách thay bằng màu lục tươi

🡪 **Shape1.Brush.Color:=ClMaroon; //** Shape1 bị mờ với màu đỏ tối

🡪 **Shape2.Brush.Color:=ClOlive; //** Shape2 bị mờ với màu vàng tối

Nhóm màu sắc độ tối và nhóm màu sắc độ sáng

|  |  |
| --- | --- |
|  |  |

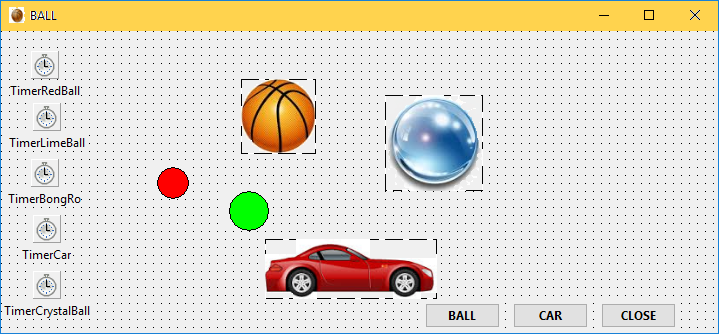
## PHẦN 25 – LẬP TRÌNH MÔ PHỎNG CÁC CHUYỂN ĐỘNG CƠ HỌC

**Ý tưởng phát triển**:

- Mô phỏng **chuyển động cơ học** & tạo **giáo cụ trực quan** trong bộ môn **Vật lý**

**Sử dụng các lớp đối tượng**

- Trong đối tượng Ball có 2 dạng lớp đối tượng là **TImage** và **TShape**



**Thuật toán** :

- Từng đối tượng chuyển động phải gắn liền với **đối tượng Timer**

|  |  |
| --- | --- |
|  | Là đối tượng **ImageBongRo** gắn liền với đối tượng **TimerBongRo** |
|  | Là đối tượng **ImageCrystalBall** gắn liền với đối tượng **TimerCrystalBall** |
|  | Là đối tượng **ShapeRedBall** gắn liền với đối tượng **TimerRedBall** |
|  | Là đối tượng **ShapeLimeBall** gắn liền với đối tượng **TimerLimeBal**l |
|  | Là đối tượng **ImageCar** gắn liền với đối **tượng TimerCar** |

- Đối tượng **Ball** chỉ xuất hiện vá chuyển động khi **Ball button** được click

- Đối tượng **Car** chỉ xuất hiện vá chuyển động khi **Car button** được click

- Ngầm định là chưa có button nào được nhấn , nên chưa có đối tượng nào xuất hiện , có nghĩa là bỏ đánh dấu thuộc tính **Visible** của từng đối tượng

|  |  |
| --- | --- |
| - Khi **Ball button** được click thì chỉ các đối tượng **Ball** được phép xuất hiện | **ShowBall:=True**; và **ShapeRedBall.Visible:=ShowBall;** |
| - Còn đối tương **Car** không được phép xuất hiện | **Showcar:=False**; và **ImageCar.Visible:=ShowCar;** |
| . Đồng thời các đối tượng Timer của đối tượng các loại bóng khác được phép có hiển thị | **ShapeRedBall.Visible:=ShowBall;ShapeLimeBall.Visible:=ShowBall; ImageBongRo.Visible:=ShowBall;**  **TimerRedBall.Enabled:=True; TimerLimeBall.Enabled:=True; TimerBongRo.Enabled:=True;**  **TimerCrystalBall.Enabled:=True;** |

- Các đối tượng còn lại thực hiện tương tự

- Để tạo hiệu ứng di chuyển phải sử dụng thuộc tính **Top** và **Left** trong các đối tượng **TShape** và **Timage**

- **Top** chứa giá trị là tọa độ cạnh trên của đối tượng khi cạnh trên của đối tượng trùng với cạnh :

🡪 Trên của cửa sổ thì giá trị Top = 0 . Với giá trị Top này thì đối tượng đổi hướng chuyển động đi xuống , nên biến UP có giá trị FALSE

🡪 Dưới của cửa sổ thì giá trị Top=Height ( Height là chiều cao cửa sổ Form là 302 ) .

- Khi tọa độ cạnh dưới của đối tượng chạm vào cạnh dưới của cửa sổ thì giá trị Top là

**Top = Height – Chiều cao của đối tượng** ( Shapex.Height và Imagex.Height – xem giá trị chiều cao đối tượng trong bảng thuộc tính Properties ) . Với giá trị Top này thì đối tượng đổi hướng chuyển động hướng lên

- Left chứa giá trị là tọa độ cạnh trái của đối tượng khi cạnh trái của đối tượng trùng với cạnh :

🡪 Trái của cửa sổ thì giá trị Left = 0 . Với giá trị Left này thì đối tượng đổi hướng chuyển động sang phải

🡪 Phải của cửa sổ thì giá trị Left=Width ( Width là chiều ngang của cửa sổ Form là 717 ) .

- Khi tọa độ cạnh phải của đối tượng chạm vào cạnh phải của cửa sổ thì giá trị Left là

**Left = Width – Chiều ngang của đối tượng** ( Shapex.Width và Imagex.Width – xem giá trị chiều ngang đối tượng trong bảng thuộc tính Properties ) . Với giá trị Left này thì đối tượng đổi hướng chuyển động sang trái

## PHẦN 26 – LẬP TRÌNH MÔ PHỎNG THIẾT BỊ LỊCH ĐIỆN TỬ, ĐỒNG HỒ

**Ý tưởng phát triển:**

- Tạo đồng hồ đa năng có thể xem ngày giờ hiện hành

- Mô phỏng đồng hồ tính giờ kiểm tra có báo thời gian còn lại mỗi khi hết 15 phút ,

- Mô phỏng đồng hồ Bấm giây

**Tạo bộ đếm bắt đầu từ 0 khi nhấn Start và ngừng đếm khi nhấn Stop và trả về 0 khi nhấn nút Clear**

|  |  |
| --- | --- |
| **Tạo đối tượng Form1**  - Thiết lập thuộc tính **Caption** là **Timer**  **Tạo đối tượng Label1**  - Vào thể **Standard** click công cụ **TLabel**  - Thiết lập các thuộc tính :  🡪 **Font > Size** 48  🡪 **Autosize** là TRUE  🡪 **Aligment** là **taCenter** để canh chính giữa  🡪 CAPTION là 0  🡪 NAME là lblDisplay |  |

**Tạo đối tượng TIMER1**

|  |  |  |
| --- | --- | --- |
| - Vào thẻ **System** click công cụ **TTimer**  - Có thể đặt **TTimer** tại bất kì vị trí nào trong khung **Label1**  - Thiết lập sự kiện **Events** chọn **Timer1Timer** bằng cách click nút …  - Lúc này trong cửa sổ **Source Editor** xuất hiện code | |  |
| **Tạo đối tượng nút START** | - Thiết lập thuộc tính : 🡪 **Name** là **btnStart** | |
| **Tạo đối tượng nút STOP** | - Thiết lập thuộc tính : 🡪 **Name** là **btnStop** | |
| **Tạo đối tượng nút CLEAR** | - Thiết lập thuộc tính : 🡪 **Name** là **btnClear** | |
| **Khai báo biến đếm**  - Đặt tên là **Counter** kiểu **Integer**  - Là biến toàn cục  - Khởi tạo giá trị là 0 |  | |

|  |  |
| --- | --- |
| **Lập trình cho đối tượng FORM1**  - Khi **Form1** xuất hiện phải vô hiệu Timer cho đến khi nút START được nhấn thì Timer mới có hiệu lực  - Click 2 lần lên vùng trống **Form** , viết code : | **Timer1.Enabled:=False;** |
| **Lập trình cho đối tượng nút START**  - Khi click nút START , Timer được kích hoạt có hiệu lực và bộ đếm sẽ hoạt động  - Click 2 lần lên nút START, viết code: | **Timer1.Enabled:=True;** |
| **Lập trình cho đối tượng nút STOP**  - Khi click nút STOP , Timer bị vô hiệu nên bộ đếm ngừng lại  - Click 2 lần lên nút STOP viết code: | **Timer1.Enabled:=False;** |
| **Lập trình cho đối tượng nút CLEAR**  - Khi click nút **CLEAR** , Timer bị vô hiệu nên bộ đếm ngừng lại  - Bộ đếm trả về giá trị khởi đầu 0 và hiển thị giá trị lên LABEL DISPLAY  - Viết code: | **Timer1.Enabled:=False; Counter:=0; lblDisplay.Caption:=IntToStr(Counter);** |

**Lập trình cho đối tượng TIMER**

|  |  |
| --- | --- |
| - **Timer** chỉ hoạt động khi **Timer1.Enabled:=True**; và chỉ ngừng hoạt động khi có chỉ thị **Timer1.Enabled:=False;**  - Khi hoạt động biến đếm tăng lên 1 sau một khoàng thời gian ngầm định trong thuộc tính **Interval**  - Giá trị **1000** trong **Interval** tương ứng với 1 giây 🡪 Có thể viết code ứng dụng tạo ra đồng hồ điện tử tính thời gian thực hiện một công việc nào đó |  |

|  |  |
| --- | --- |
| - **Double click** lên **Timer** viết code: | **Counter:=Counter+1;** // hoặc Inc(Counter,1)  **lblDisplay.Caption:=IntToStr(Counter);** |

|  |  |
| --- | --- |
| Các đối tượng sử dụng **Combobox , Datetimeflicker , Timer** |  |

## PHẦN 27 - MINH HỌA MENU & ỨNG DỤNG

**Sử dụng đối tượng lớp TMainmenu** ( trong thẻ Standard )

- Click phải lên đối tượng MainMenu, Trình biên tập **MenuEditor** để bắt đầu tạo menu

**Thuộc Tính TMainMenu**

|  |  |
| --- | --- |
| **- BidiMode :** là thuộc tính căn lề và hướng các menu, gồm các lựa chọn như sau:  🡪 **bdLeftToRight**: Căn hướng từ trái qua phải  **🡪 bdRightToLeft**: Căn hướng từ phải qua trái  [http://3.bp.blogspot.com/-RS_T_QwVktM/VXFMaULq5II/AAAAAAAACDg/O2V-eQbRowc/s320/Ch%25C6%25B0a%2Bc%25C3%25B3%2Bt%25C3%25AAn.png](http://3.bp.blogspot.com/-RS_T_QwVktM/VXFMaULq5II/AAAAAAAACDg/O2V-eQbRowc/s1600/Ch%C6%B0a+c%C3%B3+t%C3%AAn.png)  **🡪 bdRightToLeftNoAlign**: Từ phải qua trái nhưng không căn lề.  **🡪 bdRightToLeftReadingOnly**: Từ phải qua trái, chế độ chỉ đọc  - **Name:** Là tên của đối tượng này, mặc định là **MainMenu1, MainMenu2**, ...  [http://2.bp.blogspot.com/-ZB9YsPTtPyg/VXFNrCs9WNI/AAAAAAAACDo/jVmRFNF0Ib8/s1600/Ch%25C6%25B0a%2Bc%25C3%25B3%2Bt%25C3%25AAn.png](http://2.bp.blogspot.com/-ZB9YsPTtPyg/VXFNrCs9WNI/AAAAAAAACDo/jVmRFNF0Ib8/s1600/Ch%C6%B0a+c%C3%B3+t%C3%AAn.png)  **- ParentBidiMode :**  🡪 Thuộc tính này chỉ có hai giá trị **True** / **False** tương ứng **Có / Không**.  🡪 Thuộc tính này tạm hiểu là có đặt chê độ **BidiMode** cho menu cấp trên hay không | [http://1.bp.blogspot.com/-0BoG46GJAn8/VXFJpl71lBI/AAAAAAAACDE/R-YBpJJkeWw/s1600/Ch%25C6%25B0a%2Bc%25C3%25B3%2Bt%25C3%25AAn.png](http://1.bp.blogspot.com/-0BoG46GJAn8/VXFJpl71lBI/AAAAAAAACDE/R-YBpJJkeWw/s1600/Ch%C6%B0a+c%C3%B3+t%C3%AAn.png) |

**Thuộc Tính Của MenuItem hay SubMenu**

.

|  |  |
| --- | --- |
| **AutoCheck :** quy định mục đó có tự động đánh dấu hay không.  **Bitmap:** Là những icon xuất hiện trên MenuItem:  - Khuyến nghị dùng **icon 16x16**, **PNG**.  **Caption: M**ặc định hiện ra là **New Item1, New Item2**, ...  **Checked**Thiết lập MenuItem này có được **đánh dấu check** hay chưa , tương đương giá trị **True / False** | [http://2.bp.blogspot.com/-MLlIVuFMSHY/VXFSvGE8sLI/AAAAAAAACEE/8974nfLiiqQ/s1600/Ch%25C6%25B0a%2Bc%25C3%25B3%2Bt%25C3%25AAn.png](http://2.bp.blogspot.com/-MLlIVuFMSHY/VXFSvGE8sLI/AAAAAAAACEE/8974nfLiiqQ/s1600/Ch%C6%B0a+c%C3%B3+t%C3%AAn.png) |

|  |  |
| --- | --- |
| **Default** Thiêt lập MenuItem về mặc định.  **Enabled** Thiết lập MenuItem này có cho sử dụng hay không, kích hoạt hay không kích hoạt. Nếu bạn chọn False thì Item đó vẫn hiện nhưng không cho thao tác được.  Mục này rất hữu ích trong khi viết phần mềm phân quyền.  **Hint**Hiển thị gợi ý khi rê chuột vào item.  **Shortcut** Tạo tổ hợp phím truy cập nhanh.  **Visible**Có hai giá trị **True / False tương ứng với** khả dụng **hay** không khả dụng. | [http://1.bp.blogspot.com/-nUv4l2UpQFQ/VXFR57LwciI/AAAAAAAACD8/ETJMUzXnQyU/s400/Ch%25C6%25B0a%2Bc%25C3%25B3%2Bt%25C3%25AAn.png](http://1.bp.blogspot.com/-nUv4l2UpQFQ/VXFR57LwciI/AAAAAAAACD8/ETJMUzXnQyU/s1600/Ch%C6%B0a+c%C3%B3+t%C3%AAn.png) |

## PHẦN 28 - MINH HỌA LẬP TRÌNH TẠO HỆ THỐNG MENU NGANG

**Ý tưởng phát triển**

- Trong một hệ thống chương trình lớn có nhiều chức năng, khi đó người ta tổ chức một phân hệ các chức năng dưới dạng hệ thống menu

- Nhờ hệ thống menu mà có thể tạo ra từng nhóm chức năng một cách khoa học, dễ nhớ, thân thiện với người dùng

### A. Menu ngang

**Tạo hệ thống Menu gồm :**

|  |  |
| --- | --- |
| - **Menu Ngang** : **File , Edit , View**  - **Menu dọc** hay **Sub Menu**  **File 🡪 New**  **Open**  **Quit**  **Edit 🡪 Undo**  **Copy**  **Paste**  **Delete** |  |

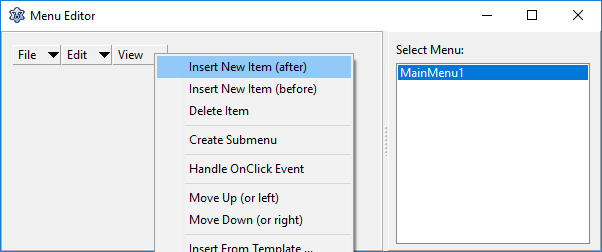
**1. Tạo menu ngang :**

|  |  |
| --- | --- |
| - Trong thẻ **Standard** , click chọn đối tượng lớp **TMainmenu**  - Click lên **Form** để đặt vào đối tượng **MainMenu1**  - Click phải lên đối tượng **MainMenu1**, chọn **Trình biên tập** **Menu Editor** |  |

|  |  |
| --- | --- |
| - Click **MenuItem1** trong Trình biện tập **Menu Editor**  - Click chức năng **+ Add menu Item** để tạo menu ngang  - Click chức năng **+ Add submenu** để tạo menu con ( **submenu** ) hay menu dọc  - Để đặt tên cho menu đang chọn, gõ tên menu trong thuộc tính **Caption** ví dụ **File**  - Trong thuộc tính **Name** của menu File gõ **mnFile**  - Double click lên tên menu để viết code cho đối tượng này  Ví dụ : **Memo1.Lines.Add(‘ menu File đã chọn’);**  - Các menu còn lại thực hiện tương tự |  |
| - Trình biên tập **Menu Editor** cho phép tạo menu con trong menu con |  |

**3. Tạo đối tượng khung Memo :**

- Click đối tượng lớp **TMemo** đặt lên Form



### B. Quick menu

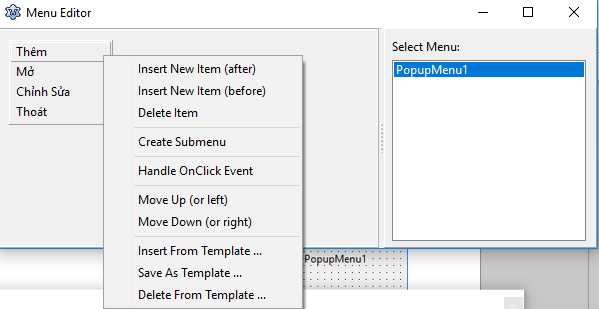
|  |  |
| --- | --- |
| - Menu xuất hiện khi click phải lên đối tượng |  |

**Thực hiện tạo menu:**

|  |  |
| --- | --- |
| - Vào thẻ **Standard** :  🡪 Click biểu tượng công cụ **TpopupMenu** , sau đó click lên bất kì vị trí nào trong form |  |
| 🡪 Click biểu tượng công cụ **TLabel** , click lên form , vào thuộc tính **Caption** xóa từ **Label1** và gõ “ **Hãy click phải lên vị trí bất kỳ trong cửa sổ** “ |  |

**Thiết lập Popup Menu**

|  |  |
| --- | --- |
| - Click phải lên **PopupMenu1** , chọn **Trình Soạn Thảo Menu Editor**  - Click MenuItem1 trong cửa sổ soạn thảo , Vào thuộc tính **Caption** sửa thành **Thêm**  - Click phải lên menu **Thêm** , chọn **Insert New Item ( after )** , lần lượt tạo 3 menu kế tiếp : **Mở - Chỉnh Sửa – Thoát** |  |



|  |  |
| --- | --- |
| Double click menu **THOÁT** chèn lệnh **Close** |  |

Liên kết **Form** và **Popup Menu**

|  |  |
| --- | --- |
| - Click lên FORM  - Vào thuộc tính **PopupMenu** sửa thành **PopupMenu1** |  |

- Vào **File > Save As** lưu với tên **MenuPopup** nhấn ENTER

Để sử dụng **TPopup**, kéo đối tượng trên Thẻ **Standard**, thả vào **Form** đang thiết kế:

|  |  |
| --- | --- |
| <http://4.bp.blogspot.com/-juVdhMWhkJs/Ve0KTQTz5UI/AAAAAAAACZw/n31_6frcSaw/s1600/tpopupmenu.png>  Có thể thay đổi các thuộc tính trong cửa sổ Kiểm tra đối tượng.  Khi kéo thả thì một đối tượng mới xuất hiện, tên mặc định là: **PopupMenu1** Đổi tên là **popup** | **[http://1.bp.blogspot.com/-cCOczDf_xaE/Ve0NVKmurVI/AAAAAAAACaQ/8aYKU5F14SI/s320/Untitled.png](http://1.bp.blogspot.com/-cCOczDf_xaE/Ve0NVKmurVI/AAAAAAAACaQ/8aYKU5F14SI/s1600/Untitled.png)** |

**Thuộc tính TPopup - Property**

**1 Bidimode**

BidiMode là thuộc tính chỉnh căn lề và hướng của các Menu, gồm các lựa chọn như sau:

**- bdLeftToRight**: Căn hướng từ bên trái qua phải:

**- bdRightToLeft**: Căn hướng từ bên phải qua trái:

**- bdRightToLeftNoAlign**: Từ phải qua trái nhưng không căn lề.

**- bdRightToLeftReadingOnly**: Từ phải qua trái, chế độ chỉ đọc

**2. Items**

Là những **sub menu** hay menu con trong **PopupMenu** . Có thể thay đổi tên trong Trình biên tập **Menu Editor**.

**3. Name :** Tên để truy xuất trong chương trình:

**Sự kiện - Events**

- Chứa các phương thức sau:.

🡪 Rê chuột ,...

🡪 Đối tượng TPopup có sự kiện :

🡪 **OnPopup**: Phương thức thực hiện khi đối tượng **PopupMenu** bật

🡪 **Onclose:** Hành động khi đối tượng **PopupMenu** đóng

## PHẦN 29 - LÀM VIỆC VỚI MỘT DANH SÁCH CHỌN - COMBOBOX

**Ý tưởng phát triển**:

- Thông thường menu chính là danh sách các chức năng để chọn lựa và thường định vị ở phía trên cửa sổ trong một hệ thống menu

- Khi muốn có một danh sách chọn lựa tại bất kỳ vị trí nào trên Form thì ComboBox là thành phần được lưa chọn để sử dụng

**VD**: Trong CSDL với một danh sách chọn có 63 mục chọn lựa cho 63 Tình, Thành của Việt Nam thì nên dùng ComboBox

**Sử dụng công cụ lớp TCombobox**

|  |  |
| --- | --- |
| - Hộp danh sách xổ xuống các mục để chọn được gọi là **items**  - Trong **ComboBox**, từng mục được lưu trữ trong thuộc tính **Items**, có kiểu **TStrings**. Vì vậy, có thể gán hoặc loại bỏ các mục trong **ComboBox** | **<http://3.bp.blogspot.com/-ShjVMpS6UzQ/VXE8Dr71f3I/AAAAAAAACAU/g0w-qtcek0g/s1600/tcombobox.png>** |

**Khai báo các mục của ComboBox trực tiếp trong thẻ Thuộc tính Properties**

|  |  |
| --- | --- |
| - Click chọn đối tượng **ComboBox** trên **Form**  - Trong thẻ **Properties**, chọn thuộc tính **Items**, click nút ba chấm |  |

|  |  |
| --- | --- |
| - Soạn thảo các mục trong **Strings Editor Dialog**, có chức năng **Sort** và **Clear** |  |

**Viết code nhập các mục vào ComboBox**

- Trong trường hợp các mục của ComboBox là thay đổi thì cần lập trình để tạo các mục trong ComboBox

VD: - Nhập các mục cho **ComboBox** trong sự kiện **OnCreate** của **Form**, bằng cách nhấp vào **Form**

- Thực hiện **double click** lên Form

- Chèn code

**procedure TForm1.FormCreate(Sender: TObject);**

**begin**

**ComboBox1.Items.Clear; //Delete all existing choices**

**ComboBox1.Items.Add('Red'); //Add an choice**

**ComboBox1.Items.Add('Green');**

**ComboBox1.Items.Add('Blue');**

**ComboBox1.Items.Add('Random Color');**

**end;**

- Khi có hành động chọn mục 1 là Red sẽ trả về 0 cho thuộc tính ItemIdex, tương tự Green sẽ là 1

- **Doubleclick ComboBox** trên Form hoặc chọn sự kiện **OnChange** trong **Object Inspector** và nhấn vào **nút [...].**

- Chèn **code** ví dụ như sau: :

**procedure TForm1.ComboBox1Change(Sender: TObject);**

**begin**

**case ComboBox1.ItemIndex of //what entry (which item) has currently been chosen**

**0: Color:=clRed;**

**1: Color:=clGreen;**

**2: Color:=clBlue;**

**3: Color:=clYellow;**

**4: Color:=Random($1000000);**

**end;**

**end;**

- Như vậy sau ki chọn màu, kết quả form đổi sang màu được chọn.

|  |  |
| --- | --- |
|  |  |

**Đặt biểu tượng cho từng mục trong ComboBox**

- Đầu tiên tải một vài hình ảnh vào trong một đối tượng **TImageList** ( trong thẻ **Common Controls** )và hiện chúng ở đầu từng mục trong **ComboBox** .

- Tạo một ứng dụng ví dụ **FillComboBox**

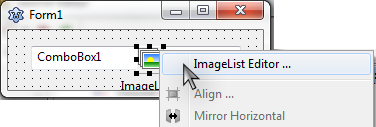
- Thay đổi thuộc tính **Style** của **ComboBox1** thành **csOwnerDrawFixed**.

- Thêm một **TImageList** từ bảng thành phần Các điều khiển chung trên form của bạn.

- Thuộc tính **Height** và **Width** **16 pixel** được đặt trước trong **ImageList1**. Để vừa vặn các hình ảnh vào **ComboBox**, thay đổi thuộc tính **ItemHeight** của **ComboBox1** thành **18** trong **Object Inspector**.

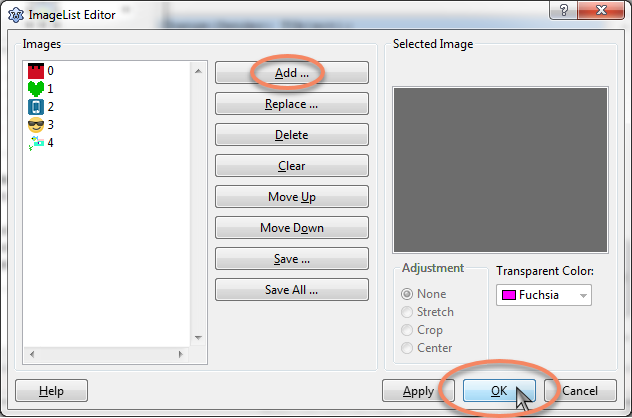
- Thêm bốn hình ảnh vào **ImageList**:

- **Double Click ImageList1** hoặc **leftclick ImageList1** và chọn **ImageList Editor** ....



- Click **Add** để thêm **4** hình ảnh **16x16 px** có sẵn từ thư mục **Lazarus\ images\**

- Bạn đã thêm bốn hình ảnh, xác nhận công việc của bạn với [OK].



- Tạo trong **Object Inspector** xử lý sự kiện cho sự kiện **OnDrawItem**, bằng cách nhấn vào **nút [...].**

Thêm mã sau vào trình xử lý:

**procedure TForm1.ComboBox1DrawItem(Control: TWinControl; Index: Integer;**

**ARect: TRect; State: TOwnerDrawState);**

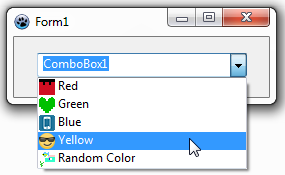
**begin**

**ComboBox1.Canvas.FillRect(ARect); //first paint normal background**

**ComboBox1.Canvas.TextRect(ARect, 20, ARect.Top, ComboBox1.Items[Index]); //paint item text**

**ImageList1.Draw(ComboBox1.Canvas, ARect.Left + 1, ARect.Top + 1, Index); //draw image according to index on canvas**

**end;**



## PHẦN 30 - MINH HỌA LẬP TRÌNH TẠO ỨNG DỤNG DUYỆT HÌNH ẢNH

**Ý tưởng phát triển**

- Duyệt hình ảnh thường là công việc khi trình bày …, có thể thiết kế cho phù hợp, tương tác thuận lợi

**Cơ bản**:

- Thực hiện duyệt hình ảnh từ tập tin trên các thiết bị lưu trữ

- Canh chỉnh hình ảnh

- Có thể đặt cửa sổ hình ảnh ở mode tự động co giãn tùy theo kích thước hình ảnh

**Sử dụng đối tượng lớp TOpenPictureDialog**

|  |  |
| --- | --- |
| - Vào thẻ **Additional, c**hèn đối tượng lớp **TImage** lên **Form**  - Vào thẻ **Standard,** chèn đối tượng lớp **TButton**  - Vào thẻ **Dialog** Chèn đối tượng **TOpenPictureDialo**g  **Đặt thuộc tính** :  - Click đối tượng **TImage**  - Đánh dấu thuộc tính **Center** để căn chỉnh hình ảnh vào chính giữa  **Viết CODE cho sự kiện nút LOAD khi được click**  - Double click nút **LOAD**  - Chèn code sau:  a. **Chèn tự động**:  **Image1.Picture.LoadFromFile('<Tên ổ đĩa>:\< Thư mục >\< Tập tin image> ');** |  |

**b. Chèn bằng các chọn tập tin hình ảnh trên thiết bị lưu trữ**

**If OpenPictureDialog1.Execute then**

**Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);**

## PHẦN 31 - MINH HỌA LẬP TRÌNH VẼ ĐƯỜNG TRÊN FORM

- Tự động vẽ ngẫu nhiên đường thẳng ( **line** ) với **Style** bằng dấu chấm (**psdot**) với màu ngẫu nhiên

- Tọa độ cuối của đường thẳng hiện thời sẽ là tọa độ đầu đường thẳng kế tiếp ( thuật toán vẽ đường gấp khúc )

### 1. Hiệu ứng vẽ đường thẳng nối tiếp

|  |  |  |
| --- | --- | --- |
| **Tạo đối tượng** :  - Vào thẻ **System** , click công cụ **TTimer** và đặt lên **Form**  - Công cụ **TTimer** tạo ra vòng lặp vô tận với sự kiện  **Viết Code cho đối tượng TTimer** :  - Double click lên công cụ **TTimer** trên Form |  |  |

**procedure TForm1.Timer1Timer(Sender: TObject);**

**var W,H:integer;**

**begin**

**W:=random(ClientWidth); //** W lưu giữ tọa độ ngang

**H:=random(ClientHeight); //** H lưu giữ tọa độ dọc ngẫu nhiên

**With canvas do**

**begin**

**pen.Style:=psdot; //**psdot chỉ định kiểu đường là dấu chấm

**pen.Width:=2;**

**pen.Color:=RGBtoColor(random(255),random(255),random(255));**

**lineTo(W,H);**

**MoveTo(W,H);**

**end;**

**end;**

### 2. Hiệu ứng pháo bông

- Để vẽ đường thẳng ngẫu nhiên với điểm tọa độ đầu của đường thẳng là chính giữa cửa sổ FORM , còn tọa độ cuối là ngẫu nhiên

- Trong đoạn code trên chỉ cần thay đổi **MoveTo(W,H)**; trong đoạn code trên thành **MoveTo(WIDTH div 2,HEIGHT div 2)**; thì tạo ra hiệu ứng giống như pháo bông

- **(WIDTH div 2,HEIGHT div 2)** là cách tính tọa độ tâm của cửa sổ FORM

### 3. Hiệu ứng nhiều chùm pháo bông

|  |  |  |
| --- | --- | --- |
| - Để vẽ thêm chùm pháo bông thứ 2 , ta đặt thêm công cụ **TTimer** 🡪 xuất hiện đối tượng **Timer2**  - Copy code của **TIMER1** sang **TIMER2** , chỉ thay đổi **MoveTo(WIDTH div 2,HEIGHT div 2)**; thành **MoveTo(WIDTH div 4,HEIGHT div 4)**; |  |  |

## PHẦN 32 - MINH HỌA XỬ LÝ HÌNH HỌC

**Ý tưởng phát triển**:

- Tạo công cụ mô phỏng hình học trực quan phù hợp với bộ môn Toán trong giảng dạy & học tập

|  |  |
| --- | --- |
| **Sử dụng đối tượng lớp** **TShape** ( trong thẻ **Additional )** |  |

|  |  |
| --- | --- |
| - Vào thẻ, click công cụ, sau đó click lên**Form**  - Ngầm định **thuộc tính** **Shape** là **stRectangle**  - Chọn thẻ Properties, click chọn thuộc tính Shape, chọn một kiểu hình khác  - Viết code: **Shape1:=stStar;** |  |

|  |  |
| --- | --- |
| - Tô màu **Shape** vào thuộc tính **Brush**, chọn thuộc tính **Color**  - Viết code: **Shape1.Brush.Color:=clYellow;** | C:\Users\LATITUDE\AppData\Local\Temp\SNAGHTMLa6bda6.PNG |

## PHẦN 33 - BIỂU DIỄN CÁC DẠNG HÌNH HỌC VỚI SỰ KIỆN ONPAINT

### 1. Tọa độ vẽ

- Thuộc tính các tọa độ đường , hình chỉ làm việc với số nguyên.

- Để thu được giá trị nguyên, có thể dùng thủ thuật là sử dụng các phép toán **DIV**, **TRUNC**

#### a. Cách dùng toán tử DIV để thu được giá trị nguyên

**Ví dụ**: **3\*width/4** sẽ cho ra số thực , để lấy kết quả là số nguyên ta gõ **3\*width div 4**

#### b. Cách Dùng hàm TRUNC(x) để lấy giá trị là phần nguyên của x

**Ví dụ**: **trunc(5,2) = 5** , **trunc(4.9) = 5**

- Tọa độ cực đại là tọa độ góc dưới phải của Form

- Hằng **WIDTH** cho giá trị là tọa độ ngang lớn nhất của cửa sổ **Form**

- Hằng **HEIGHT** cho giá trị là tọa độ dọc lớn nhất của cửa sổ **Form**

### 2. Vẽ hình, đường bằng Phương thức Canvas

|  |  |
| --- | --- |
| - Khi gõ **Canvas** . sẽ xuất hiện khung gợi ý cho phép chọn để vẽ đường , hình  - **Canvas** về mặt ý nghĩa là để chỉ mảnh vải bố mà các họa sĩ thường dùng để vẽ lên đó  - Phạm vi Phương thức **Canvas** có thể vẽ là toàn bộ cửa sổ **Form** , có nghĩa là vẽ trong phạm vi vùng của cửa sổ **Form** là (**0,0,WIDTH,HEIGHT**)  - Dùng Phương thức **Canvas** để gọi các thủ tục vẽ : |  |

|  |  |
| --- | --- |
| **Gọi Thủ tục vẽ đường thẳng LineTo(x,y)** :  Đường thẳng nồi hai tọa độ : Tử tọa độ hiện thời đến tọa độ mới là **x,y** | **Canvas.LineTo(x,y** |
| **Gọi Thủ tục đặt con trỏ tại tọa độ x,y** | **Canvas.MoveTo(x,y)** |
| **Gọi Thủ tục định chiều dày nét vẽ, màu, kiểu đường** | **Canvas.Pen**  **Các thuộc tính đối tượng Pen :**  **-** **Witdth** định **chiều dày nét vẽ**  - **Color** định **màu bút vẽ**  **- Style** định **kiểu đường** |

### 3. Phương thức Canvas trong sự kiện OnClick

**procedure TForm1.Button1Click(Sender: TObject);**

**begin**

**canvas.Pen.width:=1;**

**canvas.pen.color:=clRed;**

**canvas.line(width div 4,height div 4,width\*3 div 4,height\*3 div 4);**

**end;**

- Nhược điểm của việc vẽ hình trong sự kiện này là khi thay đổi kích thước Form hình vẽ bị biến mất

### 4. Phương thức Canvas trong sự kiện OnPaint

|  |  |  |
| --- | --- | --- |
| - Nếu **Form** thiết lập với sự kiện ( **Event** ) **OnPaint** là **FormPaint** thì các hình vẽ luôn luôn tồn tại cho dù có thay đổi kích thước cửa sổ  **Ví dụ** :  - Đường thẳng được vẽ trong sự kiên **OnClick**  - Hình chữ nhật được vẽ trên Form với sự kiện (**Event**) **OnPaint** là **FormPaint.**  - Quan sát khi thay đổi kích thước cửa sổ thì hình chữ nhật trên Form vẫn còn , còn đường thẳng vẽ trong sự kiện khác thì biến mất |  |  |

|  |  |
| --- | --- |
| **procedure TForm1.FormPaint(Sender: TObject);**  **begin**  **canvas.Pen.width:=1;**  **canvas.pen.color:=clRed;**  **canvas.rectangle(width div 4,height div 4,width\*3 div 4,height\*3 div 4);**  **end;** |  |

## PHẦN 34 - MINH HỌA LẬP TRÌNH VẼ HÌNH & TƯƠNG TÁC BẰNG MOUSE

**Ý tưởng phát triển code**:

- Lập trình mouse trong việc thiết kế đồ họa

**Cơ bản**:

- Khi click chuột phải nhận biết được là đã click chuột trái hay click chuột phải

- Hiển thị được tọa độ điểm đã click chuột

**Sử dụng sự kiện OnMouseDown**

- Sẽ thực thi chương trình khi click chuột

- Không phân biệt khi click nút trái hay phải

**Viết code:**

**Bước 1**: Double click lện Form

**Bước 2**:

Vào thẻ **Events**, click dấu 3 chấm và chọn sự kiện **OnMouseDown** sẽ phát sinh thủ tục

**procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);**

**Bước 3**: Viết code

- Để kiểm tra nút chuột nào được click dùng **if Button=mbLeft** , **if Button=mbRight**  hoặc **if Button=mbMiddle**

- Để hiển thị tọa độ ra màn hình: **ShowMessage('Click Right Mouse'+' X = '+IntToStr(x)+' Y = '+IntToStr(Y));**

**Chạy CT, xem kết quả**

- Click trái hoặc click phải lên vị trí bất kỳ trên Form, kết quả hiển thị như trong hình

|  |  |
| --- | --- |
| **C:\Users\LATITUDE\AppData\Local\Temp\SNAGHTML38344a.PNG** |  |

**Sử dụng Phương thức Canvas**

|  |  |
| --- | --- |
| - Khi click chuột trái hoặc chuột phải lên Form thì vẽ một điểm, vẽ một đường thẳng tới , hình Ellipse và hình chữ nhật tại vị trí click chuột  **Form1.Canvas.Pen.Color:=clLime; //** Thiết lập màu đường  **Form1.canvas.ellipse(x-15,y-15,x+15,y+15);** // Vẽ hình **Ellipse**  **Form1.Canvas.Pen.Color:=clRed;** // Chọn màu vẽ là màu đỏ  **Form1.Canvas.Rectangle(x-10,y-10,x+10,y+10);** // Vẽ hình chữ nhật  **Form1.Canvas.Pen.Color:=clBlue;**  **Form1.canvas.lineto(x,y);**  **Form1.Canvas.Pen.Color:=clGreen;**  **Form1.Canvas.Pixels[x,y]; //** Vẽ điểm tại tọa độ (x,y) |  |

- Ban đầu khi chưa click chuột tọa độ X=0 và Y=0 , Để ý đường thẳng bắt đầu vẽ từ tọa độ 0,0

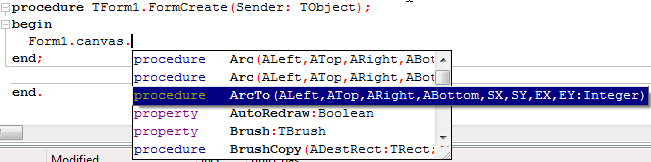
- Khi click chuột thì tọa độ X,Y sẽ thay đổi đúng vào vị trí điểm mà chuột click vào đó

- Để ý cú pháp thủ tục vẽ điểm tại tọa độ x,y là **dùng ngoặc vuông**

**form1.Canvas.Pen.Color:=clGreen; form1.Canvas.Pixels[x,y];**

- Thủ thuật để viết lại đoạn với 2 lệnh trên thành một lệnh **form1.Canvas.Pixels[x,y]:=clGreen;**

- Muốn biết thủ tục Canvas có thể vẽ được hình hoặc đường nào, thực hiện như trong hình



## PHẦN 35 – LẬP TRÌNH ĐỌC & ĐỊNH DẠNG TẬP TIN VĂN BẢN

**Ý tưởng phát triển**:

- Viết chương trình xử lý văn bản phục vụ theo yều riêng, hoặc yêu cầu công việc

**Yêu cầu thực hiện**:

|  |  |
| --- | --- |
| - Tập tin văn bản **F:\TEXT.TXT** ban đầu có nội dung & hình thức như hình bên |  |

|  |  |
| --- | --- |
| **- Viết Code**  - Xuất **F:\TEXT.TXT** theo định dạng:  🡪 4 cột phân cách các cột là dấu phảy  🡪 Độ rộng giữa các cột là 9 |  |

**Thực hiện** :

**Bước 1**: Khởi tạo **Form**

**Bước 2** : **Double Click** lên Form và tạo Code 🡪 Xóa hết nội dung trước đó trong đối tượng **Memo1**

**procedure TForm1.FormCreate(Sender: TObject);**

begin

**Memo1.Clear**;

end;

**Bước 3**: Khai báo biến File

**C1** : **Var TF:Text**; // Tương thích với FREE PASCAL

**C2** : **Var TF:TextFile**; // Khai báo trong LAZARUS

**Line, c1,c2,c3,c4:String**;

**Bước 4**: Trước khi đọc file , phải kiểm tra file có tồn tại chưa ?

- Nếu chưa, phải thông báo lỗi **if FileExists('F:\test.txt')<>True then ShowMessage('Test.txt not found')**

- nếu rồi, tiến hành thủ tục **Chuyển nội dung file vào biến file** .

- Viết Code thủ tục chép toàn bộ nội dung tập tin **TEST.TXT** vào biến file **TF** ( ghi sau IF )

**else**

**begin**

**AssignFile(TF,'F:\test.txt');**

…

- Để đọc được nội dung biến file **TF** , gõ tiếp lệnh

**Reset(TF); …**

**Bước 5**: Chép từng dòng nội dung trong biến **TF** vào một biến đặt tên là **Line** ( nên phải khai báo biến **Line** có kiểu chuỗi **String** )

- Thủ tục **Readln(TF,Line)** chép dòng nội dung hiện tại trong biến **TF** vào biến **Line** và chuyển con trỏ xuống hàng .

- Thủ tục **Readln** mỗi lần chỉ chép được 1 dòng , để chép hết 4 dòng trong tập tin phải lập đi lập lại 4 lần lệnh **Readln** . Do đó dùng vòng lặp dưới đây ( Vòng lặp ghi sau thủ tục **Reset(TF);** )

**while NOT EOF(TF) do**

**begin**

**Readln(TF,Line);**

- Mỗi dòng chứa giá trị của 4 cột 🡪 Phải khai báo 4 biến **c1,c2,c3,c4** có kiểu chuỗi **String**

- Trích giá trị thứ nhất lưu vào biến **C1** , phải tiến hành

🡪 Xác định vị trí dấu phảy trong dòng chuỗi bằng hàm **POS(‘,’,Line)**; và khai báo **biến nguyên m** để lưu vị trí kí tự dấu phảy

**Var m:integer**;

🡪 Sau lệnh **Readln …** gõ **m:=pos(‘,’,Line)**;

🡪 Giá trị bên trái dấu phảy ( tại vị trí **m-1** ) là giá trị cần chép vào biến chuỗi **c1** , gõ lệnh

**c1:=Copy(Line,1,m-1)**; // Giống hàm =**Left(Line,m-1)** trong **Excel**

🡪 Phải xóa giá trị cột thứ 1 trong biến chuỗi **Line** để chuẩn bị chép giá trị kế tiếp vào biến chuỗi **c2** , gõ :

**Delete(Line,1,m);**

🡪 **c2,c3** cách làm tương tự **c1**

**m:=pos(',',Line);**

**c2:=copy(Line,1,m-1); //giá trị cột 2**

**Delete(Line,1,m); //xóa giá trị cột 2 trong chuỗi Line**

**m:=pos(',',Line);**

**c3:=copy(Line,1,m-1); //giá trị cột 3**

**Delete(Line,1,m); //xóa giá trị cột 3 trong chuỗi Line**

🡪 Sau 3 lần lặp giá trị của Line là giá trị cột 4

**c4:=Line; //giá trị cột 4**

- Biến chuỗi **Line2** ghép các chuỗi **c1,c2,c3,c4** và **định dạng độ rộng các giá trị là 9**

**Line2:=c1+#9+c2+#9+c3+#9+c4+#9; // #9 độ rộng là 9**

- Xuất biến chuỗi **Line2** ra đối tượng memo

**Memo1.Lines.Add(Line2);**

## PHẦN 36 - LẬP TRÌNH GỌI FILE ỨNG DỤNG & MỞ FILE TÀI LIỆU

**Ý tưởng phát triển:**

- Quản lý chương trình & dữ liệu để sử dụng có hiệu quả cho yêu cầu cá nhân hoặc công việc

**Cách thực hiện**

|  |  |
| --- | --- |
| - Để gọi File 🡪 thực thi phải khai báo thư viện 🡪 **Process**  🡪 tài liệu 🡪 **LCLIntf**  - Click nút **Start** se gọi file thực thi ( **run command** ) & đồng thời mở file tài liệu ( **open document** ) của chương trình đó trong Lazarus |  |

**RunProgram := TProcess.Create(nil);**

**RunProgram.CommandLine:='C:\Program Files (x86)\Microsoft Visual Foxpro 9\vfp9.exe**

**D:\Public\Lamviec\Tracngh\kiemtra.fxp';**

**RunProgram.Execute;**

**RunProgram.Free;**

- Nút **Start** làm hai nhiệm vụ mở chương trình FOXPRO sau đó gọi một chương trình trong mội trường Foxpro là kiemtra.fxp

- Nút **HELP** sẽ mở tài liệu trợ giúp

**OpenDocument('Help.pdf');**

## PHẦN 37 - MINH HỌA LẬP TRÌNH TƯƠNG TÁC VỚI CÁC PHÍM MŨI TÊN

**Ý tưởng phát triển**:

- Khi ứng dụng nhận biết được các phím điều khiển mũi tên **Up , Down , Left , Right arrow key** thì có thể dùng các phím này để diều khiển các đối tượng trên Form

- Đây là ý tưởng này để viết code cho ứng dụng game

**Sử dụng sự kiện OnKeyDown**

-Viết code tạo thông báo nhận biết khi nhấn các phím **Up , Down , Left , Right arrow key** trên đối tượng **TEdit**

- Chọn đối tượng **TEdit**, vào thẻ **Events**, chọn sự kiện **OnKeyDown**, click nút ba chấm, chọn **Edit1KeyDown** sẽ phát sinh thủ tục

**procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);**

|  |  |
| --- | --- |
| - Khai báo thư viện **Uses …LCLType**, Viết code  - Thư viện **LCLType** chứa các hằng  **VK\_UP**  **VK\_DOWN**  VK**\_LEFT**  **VK\_RIGHT** là giá trị kiểu **Word**, giúp xác định các phím **Up,Down,Left,Right** khi được nhấn |  |

- Dùng lệnh **if (Key = VK\_DOWN), if (Key = VK\_UP), if (Key = VK\_LEFT), if (Key = VK\_RIGHT)**

- Để nhận biết tổ hợp phím **ALT+F2** dùng **if (Key = VK\_F2)** and **(ssAlt in Shift)**

- Tương tự **CTRL+F2** dùng **if (Key = VK\_F2) and (ssCtrl in Shift)**

và **if (Key = VK\_F2) and (ssShift in Shift)** kiểm tra phím **SHIFT+F2**

- Nếu đúng thì ShowMessge tên phím được nhấn

## PHẦN 38 – LẬP TRÌNH TỔNG HỢP

**Ý tưởng phát triền**

- Tổng hợp các đối tượng có trong **Trình biên dịch Lazarus** đã nêu trong các phần trên để minh họa việc phát triển ứng dụng đa chức năng



**🙡 HẾT🙣**