Analizador Léxico

Ariana Bermúdez,Ximena Bolaños, Dylan Rodríguez

Instituto Tecnológico de Costa Rica

May 26, 2017

Análisis Sintáctico

Se hizo un analizador sintáctico con la ayuda de la herramienta de Bison, para el lenguaje C y que corre en C, este analizador trabaja en conjunto con Flex, para tomar los tokens que este le otorga y revisar con las gramáticas que les sean ingresadas.

Bison

jaajaj

Código Preprocesado (Sin Pretty Print)

```
1
2  gtk / gtk . h >
3  stdlib . h >
4  string . h >
5  math . h >
6  time . h >
7  unistd . h >
8  dirent . h >
9  sys / types . h >
10 sys / stat . h >
11 libpq - fe . h >
12 locale . h >
13
14 struct
```

Código Preprocesado (Sin Pretty Print)

```
1  PARAMETROS_EXAMINER parametros ;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Código Preprocesado (Sin Pretty Print)

```
1  N_versiones = 0 ;
2  int N_estudiantes = 0 ;
3  int N_temas = 0 ;
4  int N_subtemas = 0 ;
5  int N_ajustes = 0 ;
6  long double Nota_minima , Nota_maxima ;
7  long double Nota_minima_ajustada , Nota_maxima_ajustada
       ;
8
9
10  int Frecuencias [  10 ] ;
11
12
13
14
15
16
17
18
```

Código Preprocesado (Sin Pretty Print)

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Código Preprocesado (Sin Pretty Print)

```
1
2
3
4  struct VERSION
5  {
6  char codigo [ 5 ] ;
7  struct PREGUNTA_VERSION
8  {
9  char codigo [ 7 ] ;
10 char respuesta ;
11 int orden_version [ 5 ] ;
12 } * preguntas ;
13 } * versiones = NULL ;
14 struct PREGUNTA
15 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  char tema [ CODIGO_TEMA_SIZE + 1 ] ;
3  char subtema [ CODIGO_SUBTEMA_SIZE + 1 ] ;
4  char autor [  101 ] ;
5  int orden_tema ;
6  int orden_subtema ;
7  char ejercicio [  7 ] ;
8  int secuencia ;
9  char pregunta [  7 ] ;
10 int buenos ;
11 int malos ;
12 int acumulado_opciones [ 5 ] ;
13 long double suma_seleccion [ 5 ] ;
14 long
```

Código Preprocesado (Sin Pretty Print)

```
1  double media_ex_1 [ 5 ] ;
2  long double media_ex_0 [ 5 ] ;
3  long double Rpb_opcion [ 5 ] ;
4  char correcta ;
5  long double previo ;
6  long double desviacion ;
7  long double porcentaje ;
8  long double suma_buenos ;
9  long double suma_malos ;
10 long double Rpb ;
11 long double alfa_sin ;
12 int flags [  16 ] ;
13 char texto_pregunta [  501 + 1 ] ;
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  ajuste ;
2  int revision_especial ;
3  int correctas_nuevas [ 5 ] ;
4  int actualizar ;
5  int excluir ;
6  int encoger ;
7  int verbatim ;
8  int header_encoger ;
9  int header_verbatim ;
10 int slide [ 5 ] ;
11 int encoger_opcion [ 5 ] ;
12 int verbatim_opcion [ 5 ] ;
13 int grupo_inicio ;
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  grupo_final ;
2  } ;
3  struct PREGUNTA * preguntas = NULL ;
4  struct PREGUNTA * resumen_tema_subtema = NULL ;
5  struct PREGUNTA * resumen_tema = NULL ;
6
7
8
9
10
11
12
13
14
15
16
17
18  int Banderas [ 6 ] ;
19 char * colores [ ] = { "green" , "blue" , "cyan" , "
```

Código Preprocesado (Sin Pretty Print)

```
1  * banderas [ ] = { "FL-verde.jpg" , "FL-azul.jpg" , "FL
       -cyan.jpg" , "FL-amarilla.jpg" , "FL-roja.jpg" , "
       fix.png" } ;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

Código Preprocesado (Sin Pretty Print)

```
1  Niveles_Discriminacion ;
2  int Niveles_Dificultad ;
3  int Frecuencia_total [ 11 ] [ 21 ] ;
4  char * Beamer_aspectratio [ ] = { "43" , "169" , "1610"
       , "149" , "141" , "54" , "32" } ;
5  PGconn * DATABASE ;
6  GtkBuilder * builder ;
7  GError * error ;
8  GtkWidget * window1 = NULL ;
9  GtkWidget * window2 = NULL ;
10 GtkWidget * window3 = NULL ;
11 GtkWidget * window4 = NULL ;
12 GtkWidget * window5 = NULL ;
13 GtkWidget * window6 = NULL ;
14 GtkWidget
```

Código Preprocesado (Sin Pretty Print)

```
1  * window7 = NULL ;
2  GtkWidget * EB_analisis = NULL ;
3  GtkWidget * FR_analisis = NULL ;
4  GtkSpinButton * SP_examen = NULL ;
5  GtkWidget * EN_descripcion = NULL ;
6  GtkWidget * EN_pre_examen = NULL ;
7  GtkWidget * EN_pre_examen_descripcion = NULL ;
8  GtkWidget * EN_esquema = NULL ;
9  GtkWidget * EN_esquema_descripcion = NULL ;
10 GtkWidget * EN_materia = NULL ;
11 GtkWidget * EN_materia_descripcion = NULL ;
12 GtkWidget * EN_institucion = NULL ;
13 GtkWidget * EN_escuela = NULL ;
14 GtkWidget
```

Código Preprocesado (Sin Pretty Print)

```
1  * EN_programa = NULL ;
2  GtkWidget * EN_profesor = NULL ;
3  GtkWidget * EN_fecha = NULL ;
4  GtkWidget * EN_N_preguntas = NULL ;
5  GtkWidget * EN_N_versiones = NULL ;
6  GtkWidget * EN_N_estudiantes = NULL ;
7  GtkWidget * EB_prediccion = NULL ;
8  GtkWidget * FR_prediccion = NULL ;
9  GtkWidget * EB_real = NULL ;
10 GtkWidget * FR_real = NULL ;
11 GtkWidget * EB_grafico = NULL ;
12 GtkWidget * FR_grafico = NULL ;
13 GtkToggleButton * CK_smooth = NULL ;
14 GtkWidget
```

Código Preprocesado (Sin Pretty Print)

```
1  * EB_preguntas = NULL ;
2  GtkWidget * FR_preguntas = NULL ;
3  GtkWidget * EB_ajustes = NULL ;
4  GtkWidget * FR_ajustes = NULL ;
5  GtkWidget * EB_formato = NULL ;
6  GtkWidget * FR_formato = NULL ;
7  GtkWidget * FR_procesado = NULL ;
8  GtkWidget * FR_botones = NULL ;
9  GtkWidget * FR_pregunta_actual = NULL ;
10 GtkWidget * SC_preguntas = NULL ;
11 GtkTextView * TV_pregunta = NULL ;
12 GtkTextBuffer * buffer_TV_pregunta ;
13 GtkComboBox * CB_ajuste = NULL ;
14 GtkToggleButton
```

Código Preprocesado (Sin Pretty Print)

```
1  * TG_A = NULL ;
2  GtkToggleButton * TG_B = NULL ;
3  GtkToggleButton * TG_C = NULL ;
4  GtkToggleButton * TG_D = NULL ;
5  GtkToggleButton * TG_E = NULL ;
6  GtkToggleButton * CK_no_actualiza = NULL ;
7  GtkToggleButton * CK_excluir = NULL ;
8  GtkToggleButton * CK_encoger = NULL ;
9  GtkToggleButton * CK_verbatim = NULL ;
10 GtkToggleButton * CK_header_encoger = NULL ;
11 GtkToggleButton * CK_header_verbatim = NULL ;
12 GtkToggleButton * CK_slide [ 5 ] ;
13 GtkToggleButton * CK_encoger_opcion [ 5 ] ;
14 GtkToggleButton
```

Código Preprocesado (Sin Pretty Print)

```
1  * CK_verbatim_opcion [ 5 ] ;
2  GtkWidget * EB_beamer = NULL ;
3  GtkWidget * FR_beamer = NULL ;
4  GtkComboBox * CB_estilo = NULL ;
5  GtkComboBox * CB_color = NULL ;
6  GtkComboBox * CB_font = NULL ;
7  GtkComboBox * CB_size = NULL ;
8  GtkComboBox * CB_aspecto = NULL ;
9  GtkToggleButton * CK_general = NULL ;
10 GtkToggleButton * CK_sin_banderas = NULL ;
11 GtkWidget * EN_media_prediccion = NULL ;
12 GtkWidget * EN_desviacion_prediccion = NULL ;
13 GtkWidget * EN_alfa_prediccion = NULL ;
14 GtkWidget
```

Código Preprocesado (Sin Pretty Print)

```
1  * EN_Rpb_prediccion = NULL ;
2  GtkWidget * EN_media_real = NULL ;
3  GtkWidget * EN_desviacion_real = NULL ;
4  GtkWidget * EN_alfa_real = NULL ;
5  GtkWidget * EN_Rpb_real = NULL ;
6  GtkWidget * BN_undo = NULL ;
7  GtkWidget * BN_print = NULL ;
8  GtkWidget * BN_slides = NULL ;
9  GtkWidget * BN_save = NULL ;
10 GtkWidget * BN_terminar = NULL ;
11 GtkWidget * BN_ok = NULL ;
12 GtkWidget * BN_error_encontrado_Beamer = NULL ;
13 GtkWidget * EB_generando_beamer = NULL ;
14 GtkWidget
```

Código Preprocesado (Sin Pretty Print)

```
1  * FR_generando_beamer = NULL ;
2  GtkWidget * EB_generando_analisis = NULL ;
3  GtkWidget * FR_generando_analisis = NULL ;
4  GtkWidget * FR_error_Beamer = NULL ;
5  GtkWidget * BN_confirma_revision = NULL ;
6  GtkWidget * BN_cancela_revision = NULL ;
7  GtkWidget * EB_revisando_beamer = NULL ;
8  GtkWidget * FR_revisando_beamer = NULL ;
9  GtkWidget * FR_error_encontrado_Beamer = NULL ;
10 GtkLabel * LB_error_encontrado_Beamer = NULL ;
11 GtkWidget * PB_analisis = NULL ;
12 GtkWidget * PB_beamer = NULL ;
13 GtkWidget * PB_revisando_beamer = NULL ;
14 GtkSpinButton
```

Código Preprocesado (Sin Pretty Print)

```
1  * SP_resolucion = NULL ;
2  GtkSpinButton * SP_color = NULL ;
3  GtkSpinButton * SP_rotacion = NULL ;
4  void Actualice_Estadisticas ( int buena , char *
       ejercicio , int secuencia , long double Nota , char
        respuesta , int opcion_original [ 5 ] ) ;
5  void Actualiza_Porcentajes ( ) ;
6  void Analisis_General ( FILE * Archivo_Latex , long
       double alfa , long double Rpb , int
       Beamer_o_reporte ) ;
7  void Analiza_Ajuste ( FILE * Archivo_Latex , struct
       PREGUNTA item , int modo ) ;
8  void Analiza_Banderas ( FILE * Archivo_Latex , struct
       PREGUNTA item , int beamer , int N_flags , int i ,
       char * Descripcion ) ;
9  void Asigna_Banderas ( ) ;
10 void Beamer_Cierre ( FILE * Archivo_Latex ) ;
11 void Beamer_Cover ( FILE * Archivo_Latex ) ;
12 void Beamer_Dificultad_vs_Discriminacion ( FILE *
```

Código Preprocesado (Sin Pretty Print)

```
1 Beamer_Gracias ( FILE * Archivo_Latex ) ;
2 void Beamer_Grafico_Pastel ( FILE * Archivo_Latex ) ;
3 void Beamer_Datos_Generales ( FILE * Archivo_Latex ,
    gchar * institucion , gchar * escuela , gchar *
    programa , gchar * materia_descripcion , gchar *
    profesor , gchar * descripcion , gchar * fecha ,
    char * codigo , long double media_real , long
    double desviacion_real , long double alfa , long
    double Rpb ) ;
4 void Beamer_Histograma_Notas ( FILE * Archivo_Latex ,
    long double media_real , long double
    desviacion_real , long double media_prediccion ,
    long double desviacion_prediccion ) ;
5 void Beamer_Histograma_Temas ( FILE * Archivo_Latex ) ;
6 void Beamer_Preamble ( FILE * Archivo_Latex , int
    aspecto , gchar * size , gchar * estilo , gchar *
    color , gchar * font , gchar * materia_descripcion
    , gchar * descripcion , gchar * profesor , gchar *
    programa , gchar * escuela , gchar * institucion ,
```

Código Preprocesado (Sin Pretty Print)

```
1  Calcula_Notas ( char * version , char * respuestas ,
       int * n_buenas , int * n_ajustado , int *
       m_ajustado ) ;
2  void Calcular_Tabla ( ) ;
3  void Cambia_Pregunta ( ) ;
4  void Cambio_A ( GtkWidget * widget , gpointer user_data
       ) ;
5  void Cambio_Ajuste ( GtkWidget * widget , gpointer
       user_data ) ;
6  void Cambio_B ( GtkWidget * widget , gpointer user_data
       ) ;
7  void Cambio_C ( GtkWidget * widget , gpointer user_data
       ) ;
8  void Cambio_D ( GtkWidget * widget , gpointer user_data
       ) ;
9  void Cambio_E ( GtkWidget * widget , gpointer user_data
       ) ;
10 void Cambio_encoger ( GtkWidget * widget , gpointer
       user_data ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  Cambio_header_encoger ( GtkWidget * widget , gpointer
       user_data ) ;
2  void Cambio_header_verbatim ( GtkWidget * widget ,
       gpointer user_data ) ;
3  void Cambio_no_actualizar ( GtkWidget * widget ,
       gpointer user_data ) ;
4  void Cambio_slide ( int i ) ;
5  void Cambio_verbatim ( GtkWidget * widget , gpointer
       user_data ) ;
6  void Cambio_verbatim_opcion ( int i ) ;
7  void Carga_preguntas_examen ( ) ;
8  long double CDF ( long double X , long double Media ,
       long double Desv ) ;
9  void Color_ajustes ( int k ) ;
10 void Color_Fila ( FILE * Archivo_Latex , int flags [
       16 ] ) ;
11 void colores_pastel ( FILE * Archivo_Latex ) ;
12 void Connect_Widgets ( ) ;
13 void Construye_versiones ( char * examen ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  Continuar_banderas ( FILE * Archivo_Latex , int i ,
       char * Descripcion ) ;
2  void Crea_archivo_datos_pastel ( int * Empates ) ;
3  void Dificultad_vs_Discriminacion ( ) ;
4  void Establece_Directorio ( char * Directorio , gchar *
       materia , char * year , char * month , char * day
       ) ;
5  void Fin_de_Programa ( GtkWidget * widget , gpointer
       user_data ) ;
6  void Fin_Ventana ( GtkWidget * widget , gpointer
       user_data ) ;
7  void Genera_Beamer ( ) ;
8  int Genera_Beamer_reducido ( ) ;
9  void Graba_Ajustes ( ) ;
10 void Imprime_Opcion ( FILE * Archivo_Latex , PGresult *
       res , long double Porcentaje , int pregunta , int
       opcion ) ;
11 void Imprime_Opcion_Beamer ( FILE * Archivo_Latex ,
       PGresult * res , long double Porcentaje , int
```

Código Preprocesado (Sin Pretty Print)

```
1  Imprime_Reporte ( ) ;
2  void Interface_Coloring ( ) ;
3  void Inicializa_Tabla_estadisticas ( ) ;
4  void Init_Fields ( ) ;
5  void Lista_de_Notas ( FILE * Archivo_Latex ) ;
6  void Lista_de_Preguntas ( FILE * Archivo_Latex ,
       GtkWidget * PB , long double base , long double
       limite ) ;
7  void Lista_de_Preguntas_Beamer ( FILE * Archivo_Latex ,
        GtkWidget * PB , long double base , long double
       limite ) ;
8  int main ( int argc , char * argv [ ] ) ;
9  void Marca_agua_ajuste ( FILE * Archivo_Latex , int i )
        ;
10 void on_BN_ok_clicked ( GtkWidget * widget , gpointer
       user_data ) ;
11 void on_BN_print_clicked ( GtkWidget * widget ,
       gpointer user_data ) ;
12 void on_BN_save_clicked ( GtkWidget * widget , gpointer
```

Código Preprocesado (Sin Pretty Print)

```
1  on_BN_terminar_clicked ( GtkWidget * widget , gpointer
       user_data ) ;
2  void on_BN_undo_clicked ( GtkWidget * widget , gpointer
        user_data ) ;
3  void on_CB_ajuste_changed ( GtkWidget * widget ,
       gpointer user_data ) ;
4  void on_CK_encoger_A_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
5  void on_CK_encoger_B_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
6  void on_CK_encoger_C_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
7  void on_CK_encoger_D_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
8  void on_CK_encoger_E_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
9  void on_CK_encoger_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
10 void on_CK_excluir_toggled ( GtkWidget * widget ,
```

Código Preprocesado (Sin Pretty Print)

```
1  on_CK_slide_A_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
2  void on_CK_slide_B_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
3  void on_CK_slide_C_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
4  void on_CK_slide_D_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
5  void on_CK_slide_E_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
6  void on_CK_verbatim_A_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
7  void on_CK_verbatim_B_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
8  void on_CK_verbatim_C_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
9  void on_CK_verbatim_D_toggled ( GtkWidget * widget ,
       gpointer user_data ) ;
10 void on_CK_verbatim_E_toggled ( GtkWidget * widget ,
```

Código Preprocesado (Sin Pretty Print)

```
1  on_SP_examen_activate ( GtkWidget * widget , gpointer
       user_data ) ;
2  void on_TG_A_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
3  void on_TG_B_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
4  void on_TG_C_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
5  void on_TG_D_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
6  void on_TG_E_toggled ( GtkWidget * widget , gpointer
       user_data ) ;
7  void on_WN_ex4010_destroy ( GtkWidget * widget ,
       gpointer user_data ) ;
8  void Prepara_Grafico_Normal ( long double media , long
       double desviacion , long double media_pred , long
       double desviacion_pred , long double width ) ;
9  void Prepara_Grafico_Pastel ( FILE * Archivo_Latex ) ;
10 void Prepara_Histograma_Notas ( ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  Read_Only_Fields ( ) ;
2  void Quita_espacios ( char * hilera ) ;
3  void Resumen_de_Banderas ( FILE * Archivo_Latex , int
       modo ) ;
4  void Update_PB ( GtkWidget * PB , long double R ) ;
5  void Tabla_Datos_Generales ( FILE * Archivo_Latex ,
       char * institucion , char * escuela , char *
       programa , char * materia_descripcion , char *
       profesor , char * descripcion , char * fecha , char
        * codigo , long double media_real , long double
       desviacion_real , long double alfa , long double
       Rpb , int Beamer_o_reporte ) ;
6  int main ( int argc , char * argv [ ] )
7  {
8  DATABASE = EX_connect_data_base ( ) ;
9  if ( PQstatus ( DATABASE ) == CONNECTION_BAD )
10 {
11 fprintf ( stderr , "Connection to database failed.\n" )
       ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  else
3  {
4  gtk_init ( & argc , & argv ) ;
5  setlocale ( LC_NUMERIC , "en_US.UTF-8" ) ;
6  carga_parametros_EXAMINER ( & parametros , DATABASE ) ;
7  builder = gtk_builder_new ( ) ;
8  if ( ! gtk_builder_add_from_file ( builder , ".
       interfaces/EX4010.glade" , & error ) )
9  {
10 g_warning ( "%s\n" , error -> message ) ;
11 g_error_free ( error ) ;
12 }
13 else
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  gtk_builder_connect_signals ( builder , NULL ) ;
3  Connect_Widgets ( ) ;
4  Read_Only_Fields ( ) ;
5  Interface_Coloring ( ) ;
6  gtk_widget_show ( window1 ) ;
7  Init_Fields ( ) ;
8  gtk_main ( ) ;
9  }
10 }
11 return 0 ;
12 }
13 void Connect_Widgets ( )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  window1 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_ex4010" ) ) ;
3  window2 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_procesado" ) ) ;
4  window3 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_generando_beamer" ) ) ;
5  window4 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_generando_analisis" ) ) ;
6  window5 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_error_Beamer" ) ) ;
7  window6 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_revisando_beamer" ) ) ;
8  window7 = GTK_WIDGET ( gtk_builder_get_object ( builder
       , "WN_error_encontrado_Beamer" ) ) ;
9  EB_analisis = GTK_WIDGET ( gtk_builder_get_object (
       builder , "EB_analisis" ) ) ;
10 FR_analisis = GTK_WIDGET ( gtk_builder_get_object (
       builder , "FR_analisis" ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1 = GTK_WIDGET ( gtk_builder_get_object ( builder , "
    EN_pre_examen_descripcion" ) ) ;
2 EN_esquema = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EN_esquema" ) ) ;
3 EN_esquema_descripcion = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    EN_esquema_descripcion" ) ) ;
4 EN_materia = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EN_materia" ) ) ;
5 EN_materia_descripcion = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    EN_materia_descripcion" ) ) ;
6 EN_institucion = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EN_institucion" ) ) ;
7 EN_escuela = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EN_escuela" ) ) ;
8 EN_programa = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EN_programa" ) ) ;
9 EN_profesor = GTK_WIDGET ( gtk_builder_get_object (
```

Código Preprocesado (Sin Pretty Print)

```
1 = GTK_WIDGET ( gtk_builder_get_object ( builder , "
    EB_prediccion" ) ) ;
2 FR_prediccion = GTK_WIDGET ( gtk_builder_get_object (
    builder , "FR_prediccion" ) ) ;
3 EN_media_prediccion = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    EN_media_prediccion" ) ) ;
4 EN_desviacion_prediccion = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    EN_desviacion_prediccion" ) ) ;
5 EN_alfa_prediccion = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    EN_alfa_prediccion" ) ) ;
6 EN_Rpb_prediccion = GTK_WIDGET ( gtk_builder_get_object
     ( builder , "EN_Rpb_prediccion" ) ) ;
7 EB_real = GTK_WIDGET ( gtk_builder_get_object ( builder
     , "EB_real" ) ) ;
8 FR_real = GTK_WIDGET ( gtk_builder_get_object ( builder
     , "FR_real" ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1 = GTK_WIDGET ( gtk_builder_get_object ( builder , "
    EB_grafico" ) ) ;
2 FR_grafico = GTK_WIDGET ( gtk_builder_get_object (
    builder , "FR_grafico" ) ) ;
3 SP_resolucion = ( GtkSpinButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "SP_resolucion"
    ) ) ;
4 SP_color = ( GtkSpinButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "SP_color" ) ) ;
5 SP_rotacion = ( GtkSpinButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "SP_rotacion" )
    ) ;
6 CK_smooth = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CK_smooth" ) )
    ;
7 FR_botones = GTK_WIDGET ( gtk_builder_get_object (
    builder , "FR_botones" ) ) ;
8 EB_preguntas = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EB_preguntas" ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1 = GTK_WIDGET ( gtk_builder_get_object ( builder , "
    FR_pregunta_actual" ) ) ;
2 SC_preguntas = GTK_WIDGET ( gtk_builder_get_object (
    builder , "SC_preguntas" ) ) ;
3 TV_pregunta = ( GtkTextView * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TV_pregunta" )
    ) ;
4 buffer_TV_pregunta = gtk_text_view_get_buffer (
    TV_pregunta ) ;
5 EB_beamer = GTK_WIDGET ( gtk_builder_get_object (
    builder , "EB_beamer" ) ) ;
6 FR_beamer = GTK_WIDGET ( gtk_builder_get_object (
    builder , "FR_beamer" ) ) ;
7 CB_estilo = ( GtkComboBox * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CB_estilo" ) )
    ;
8 CB_color = ( GtkComboBox * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CB_color" ) ) ;
9 CB_font = ( GtkComboBox * ) GTK_WIDGET (
```

Código Preprocesado (Sin Pretty Print)

```
1 = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TG_A" ) ) ;
2 TG_B = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TG_B" ) ) ;
3 TG_C = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TG_C" ) ) ;
4 TG_D = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TG_D" ) ) ;
5 TG_E = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "TG_E" ) ) ;
6 CB_ajuste = ( GtkComboBox * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CB_ajuste" ) )
    ;
7 CK_no_actualiza = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CK_no_actualiza
    " ) ) ;
8 CK_excluir = ( GtkToggleButton * ) GTK_WIDGET (
    gtk_builder_get_object ( builder , "CK_excluir" ) )
     ;
```

Código Preprocesado (Sin Pretty Print)

```
1 [ 1 ] = ( GtkToggleButton * ) GTK_WIDGET (
      gtk_builder_get_object ( builder , "CK_slide_B" ) )
       ;
2 CK_slide [ 2 ] = ( GtkToggleButton * ) GTK_WIDGET (
      gtk_builder_get_object ( builder , "CK_slide_C" ) )
       ;
3 CK_slide [ 3 ] = ( GtkToggleButton * ) GTK_WIDGET (
      gtk_builder_get_object ( builder , "CK_slide_D" ) )
       ;
4 CK_slide [ 4 ] = ( GtkToggleButton * ) GTK_WIDGET (
      gtk_builder_get_object ( builder , "CK_slide_E" ) )
       ;
5 CK_encoger_opcion [ 0 ] = ( GtkToggleButton * )
     GTK_WIDGET ( gtk_builder_get_object ( builder , "
     CK_encoger_A" ) ) ;
6 CK_encoger_opcion [ 1 ] = ( GtkToggleButton * )
     GTK_WIDGET ( gtk_builder_get_object ( builder , "
     CK_encoger_B" ) ) ;
7 CK_encoger_opcion [ 2 ] = ( GtkToggleButton * )
```

Código Preprocesado (Sin Pretty Print)

```
1 [ 4 ] = ( GtkToggleButton * ) GTK_WIDGET (
      gtk_builder_get_object ( builder , "CK_verbatim_E"
      ) ) ;
2 EB_generando_beamer = GTK_WIDGET (
      gtk_builder_get_object ( builder , "
      EB_generando_beamer" ) ) ;
3 FR_generando_beamer = GTK_WIDGET (
      gtk_builder_get_object ( builder , "
      FR_generando_beamer" ) ) ;
4 EB_generando_analisis = GTK_WIDGET (
      gtk_builder_get_object ( builder , "
      EB_generando_analisis" ) ) ;
5 FR_generando_analisis = GTK_WIDGET (
      gtk_builder_get_object ( builder , "
      FR_generando_analisis" ) ) ;
6 FR_error_Beamer = GTK_WIDGET ( gtk_builder_get_object (
       builder , "FR_error_Beamer" ) ) ;
7 EB_revisando_beamer = GTK_WIDGET (
      gtk_builder_get_object ( builder , "
```

Código Preprocesado (Sin Pretty Print)

```
1 = GTK_WIDGET ( gtk_builder_get_object ( builder , "
    BN_undo" ) ) ;
2 BN_terminar = GTK_WIDGET ( gtk_builder_get_object (
    builder , "BN_terminar" ) ) ;
3 BN_ok = GTK_WIDGET ( gtk_builder_get_object ( builder ,
    "BN_ok" ) ) ;
4 PB_analisis = GTK_WIDGET ( gtk_builder_get_object (
    builder , "PB_analisis" ) ) ;
5 PB_beamer = GTK_WIDGET ( gtk_builder_get_object (
    builder , "PB_beamer" ) ) ;
6 PB_revisando_beamer = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    PB_revisando_beamer" ) ) ;
7 BN_confirma_revision = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    BN_confirma_revision" ) ) ;
8 BN_cancela_revision = GTK_WIDGET (
    gtk_builder_get_object ( builder , "
    BN_cancela_revision" ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EN_pre_examen , FALSE ) ;
2  gtk_widget_set_can_focus ( EN_pre_examen_descripcion ,
       FALSE ) ;
3  gtk_widget_set_can_focus ( EN_esquema , FALSE ) ;
4  gtk_widget_set_can_focus ( EN_esquema_descripcion ,
       FALSE ) ;
5  gtk_widget_set_can_focus ( EN_materia , FALSE ) ;
6  gtk_widget_set_can_focus ( EN_materia_descripcion ,
       FALSE ) ;
7  gtk_widget_set_can_focus ( EN_fecha , FALSE ) ;
8  gtk_widget_set_can_focus ( EN_N_preguntas , FALSE ) ;
9  gtk_widget_set_can_focus ( EN_N_versiones , FALSE ) ;
10 gtk_widget_set_can_focus ( EN_N_estudiantes , FALSE ) ;
11 gtk_widget_set_can_focus ( EN_profesor , FALSE ) ;
12 gtk_widget_set_can_focus ( EN_institucion , FALSE ) ;
13 gtk_widget_set_can_focus ( EN_escuela , FALSE ) ;
14 gtk_widget_set_can_focus
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EN_programa , FALSE ) ;
2  gtk_widget_set_can_focus ( EN_media_prediccion , FALSE
       ) ;
3  gtk_widget_set_can_focus ( EN_desviacion_prediccion ,
       FALSE ) ;
4  gtk_widget_set_can_focus ( EN_alfa_prediccion , FALSE )
        ;
5  gtk_widget_set_can_focus ( EN_Rpb_prediccion , FALSE )
       ;
6  gtk_widget_set_can_focus ( EN_media_real , FALSE ) ;
7  gtk_widget_set_can_focus ( EN_desviacion_real , FALSE )
        ;
8  gtk_widget_set_can_focus ( EN_alfa_real , FALSE ) ;
9  gtk_widget_set_can_focus ( EN_Rpb_real , FALSE ) ;
10 gtk_widget_set_can_focus ( GTK_WIDGET ( TV_pregunta ) ,
        FALSE ) ;
11 }
12 void Interface_Coloring ( )
13 {
```

Código Preprocesado (Sin Pretty Print)

```
1  color ;
2  gdk_color_parse ( MAIN_WINDOW , & color ) ;
3  gtk_widget_modify_bg ( window1 , GTK_STATE_NORMAL , &
       color ) ;
4  gdk_color_parse ( MAIN_AREA , & color ) ;
5  gtk_widget_modify_bg ( EB_analisis , GTK_STATE_NORMAL ,
       & color ) ;
6  gdk_color_parse ( SECONDARY_AREA , & color ) ;
7  gtk_widget_modify_bg ( EB_ajustes , GTK_STATE_NORMAL ,
       & color ) ;
8  gtk_widget_modify_bg ( EB_formato , GTK_STATE_NORMAL ,
       & color ) ;
9  gdk_color_parse ( THIRD_AREA , & color ) ;
10 gtk_widget_modify_bg ( EB_preguntas , GTK_STATE_NORMAL
       , & color ) ;
11 gtk_widget_modify_bg ( EB_beamer , GTK_STATE_NORMAL , &
       color ) ;
12 gtk_widget_modify_bg ( EB_grafico , GTK_STATE_NORMAL ,
       & color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EB_prediccion , GTK_STATE_NORMAL , & color ) ;
2  gdk_color_parse ( SPECIAL_AREA_2 , & color ) ;
3  gtk_widget_modify_bg ( EB_real , GTK_STATE_NORMAL , &
       color ) ;
4  gdk_color_parse ( IMPORTANT_WINDOW , & color ) ;
5  gtk_widget_modify_bg ( window2 , GTK_STATE_NORMAL , &
       color ) ;
6  gtk_widget_modify_bg ( window5 , GTK_STATE_NORMAL , &
       color ) ;
7  gtk_widget_modify_bg ( window7 , GTK_STATE_NORMAL , &
       color ) ;
8  gdk_color_parse ( IMPORTANT_FR , & color ) ;
9  gtk_widget_modify_bg ( FR_procesado , GTK_STATE_NORMAL
       , & color ) ;
10 gtk_widget_modify_bg ( FR_error_Beamer ,
       GTK_STATE_NORMAL , & color ) ;
11 gtk_widget_modify_bg ( FR_error_encontrado_Beamer ,
       GTK_STATE_NORMAL , & color ) ;
12 gdk_color_parse ( STANDARD_FRAME , & color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( FR_preguntas , GTK_STATE_NORMAL , & color ) ;
2  gtk_widget_modify_bg ( FR_beamer , GTK_STATE_NORMAL , &
       color ) ;
3  gtk_widget_modify_bg ( FR_botones , GTK_STATE_NORMAL ,
       & color ) ;
4  gtk_widget_modify_bg ( FR_grafico , GTK_STATE_NORMAL ,
       & color ) ;
5  gtk_widget_modify_bg ( FR_generando_analisis ,
       GTK_STATE_NORMAL , & color ) ;
6  gtk_widget_modify_bg ( FR_generando_beamer ,
       GTK_STATE_NORMAL , & color ) ;
7  gtk_widget_modify_bg ( FR_revisando_beamer ,
       GTK_STATE_NORMAL , & color ) ;
8  gtk_widget_modify_bg ( FR_ajustes , GTK_STATE_NORMAL ,
       & color ) ;
9  gtk_widget_modify_bg ( FR_formato , GTK_STATE_NORMAL ,
       & color ) ;
10 gtk_widget_modify_bg ( FR_prediccion , GTK_STATE_NORMAL
       , & color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_WIDGET ( SP_examen ) , GTK_STATE_NORMAL , & color
       ) ;
2  gdk_color_parse ( BUTTON_PRELIGHT , & color ) ;
3  gtk_widget_modify_bg ( BN_save , GTK_STATE_PRELIGHT , &
       color ) ;
4  gtk_widget_modify_bg ( BN_slides , GTK_STATE_PRELIGHT ,
       & color ) ;
5  gtk_widget_modify_bg ( BN_undo , GTK_STATE_PRELIGHT , &
       color ) ;
6  gtk_widget_modify_bg ( BN_terminar , GTK_STATE_PRELIGHT
       , & color ) ;
7  gtk_widget_modify_bg ( BN_print , GTK_STATE_PRELIGHT ,
      & color ) ;
8  gtk_widget_modify_bg ( BN_cancela_revision ,
      GTK_STATE_PRELIGHT , & color ) ;
9  gtk_widget_modify_bg ( BN_error_encontrado_Beamer ,
      GTK_STATE_PRELIGHT , & color ) ;
10 gtk_widget_modify_bg ( BN_confirma_revision ,
      GTK_STATE_PRELIGHT , & color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( BN_save , GTK_STATE_ACTIVE , & color ) ;
2  gtk_widget_modify_bg ( BN_slides , GTK_STATE_ACTIVE , &
       color ) ;
3  gtk_widget_modify_bg ( BN_undo , GTK_STATE_ACTIVE , &
       color ) ;
4  gtk_widget_modify_bg ( BN_terminar , GTK_STATE_ACTIVE ,
       & color ) ;
5  gtk_widget_modify_bg ( BN_print , GTK_STATE_ACTIVE , &
       color ) ;
6  gtk_widget_modify_bg ( BN_cancela_revision ,
       GTK_STATE_ACTIVE , & color ) ;
7  gtk_widget_modify_bg ( BN_error_encontrado_Beamer ,
       GTK_STATE_ACTIVE , & color ) ;
8  gtk_widget_modify_bg ( BN_confirma_revision ,
       GTK_STATE_ACTIVE , & color ) ;
9  gtk_widget_modify_bg ( BN_cancela_revision ,
       GTK_STATE_ACTIVE , & color ) ;
10 gtk_widget_modify_bg ( BN_ok , GTK_STATE_ACTIVE , &
       color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_WIDGET ( PB_revisando_beamer ) ,
       GTK_STATE_PRELIGHT , & color ) ;
2  gdk_color_parse ( PROCESSING_WINDOW , & color ) ;
3  gtk_widget_modify_bg ( EB_generando_analisis ,
       GTK_STATE_NORMAL , & color ) ;
4  gtk_widget_modify_bg ( EB_generando_beamer ,
       GTK_STATE_NORMAL , & color ) ;
5  gtk_widget_modify_bg ( EB_revisando_beamer ,
       GTK_STATE_NORMAL , & color ) ;
6  }
7  void Init_Fields ( )
8  {
9  char Examen [  6 ] ;
10 char hilera [ 100 ] ;
11 int month , year , day ;
12 PGresult * res ;
13 int i , Last ;
14 GdkColor
```

Código Preprocesado (Sin Pretty Print)

```
1  color ;
2  res = PQEXEC ( DATABASE , "SELECT codigo_examen from
       EX_examenes order by codigo_examen DESC limit 1" )
       ;
3  Last = 0 ;
4  if ( PQntuples ( res ) )
5  {
6  Last = atoi ( PQgetvalue ( res , 0 , 0 ) ) ;
7  }
8  PQclear ( res ) ;
9  if ( Last > 1 )
10 {
11 gtk_widget_set_sensitive ( GTK_WIDGET ( SP_examen ) , 1
       ) ;
12 gtk_spin_button_set_range ( SP_examen , 1.0 , ( long
       double ) Last ) ;
13 }
14 else
```

Código Preprocesado (Sin Pretty Print)

```
1
2  {
3  gtk_widget_set_sensitive ( GTK_WIDGET ( SP_examen ) , 0
       ) ;
4  gtk_spin_button_set_range ( SP_examen , 0.0 , ( long
       double ) Last ) ;
5  }
6  gtk_spin_button_set_value ( SP_examen , Last ) ;
7  if ( preguntas ) free ( preguntas ) ;
8  preguntas = NULL ;
9  if ( resumen_tema ) free ( resumen_tema ) ;
10 resumen_tema = NULL ;
11 if ( resumen_tema_subtema ) free ( resumen_tema_subtema
       ) ;
12 resumen_tema_subtema = NULL ;
13 if ( versiones )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  for ( i = 0 ; i < N_versiones ; i ++ ) free ( versiones
       [ i ] . preguntas ) ;
3  free ( versiones ) ;
4  versiones = NULL ;
5  }
6  gtk_entry_set_text ( GTK_ENTRY ( EN_fecha ) , "\0" ) ;
7  N_preguntas = N_versiones = N_estudiantes = 0 ;
8  sprintf ( hilera , "%7d" , N_preguntas ) ;
9  gtk_entry_set_text ( GTK_ENTRY ( EN_N_preguntas ) ,
       hilera ) ;
10 gtk_entry_set_text ( GTK_ENTRY ( EN_N_versiones ) ,
       hilera ) ;
11 gtk_entry_set_text ( GTK_ENTRY ( EN_N_estudiantes ) ,
       hilera ) ;
12 gtk_entry_set_text ( GTK_ENTRY ( EN_descripcion ) , "\0
       " ) ;
13 gtk_entry_set_text ( GTK_ENTRY ( EN_pre_examen ) , "\0"
       ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_ENTRY ( EN_pre_examen_descripcion ) , "\0" ) ;
2  gtk_entry_set_text ( GTK_ENTRY ( EN_esquema ) , "\0" )
       ;
3  gtk_entry_set_text ( GTK_ENTRY ( EN_esquema_descripcion
       ) , "\0" ) ;
4  gtk_entry_set_text ( GTK_ENTRY ( EN_materia ) , "\0" )
       ;
5  gtk_entry_set_text ( GTK_ENTRY ( EN_materia_descripcion
       ) , "\0" ) ;
6  gtk_entry_set_text ( GTK_ENTRY ( EN_institucion ) , "\0
       " ) ;
7  gtk_entry_set_text ( GTK_ENTRY ( EN_escuela ) , "\0" )
       ;
8  gtk_entry_set_text ( GTK_ENTRY ( EN_programa ) , "\0" )
       ;
9  gtk_entry_set_text ( GTK_ENTRY ( EN_profesor ) , "\0" )
       ;
10 gtk_widget_set_sensitive ( GTK_WIDGET ( SP_examen ) , 1
       ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EN_esquema_descripcion , 0 ) ;
2  gtk_widget_set_sensitive ( EN_materia , 0 ) ;
3  gtk_widget_set_sensitive ( EN_materia_descripcion , 0 )
       ;
4  gtk_widget_set_sensitive ( EN_fecha , 0 ) ;
5  gtk_widget_set_sensitive ( EN_N_preguntas , 0 ) ;
6  gtk_widget_set_sensitive ( EN_N_versiones , 0 ) ;
7  gtk_widget_set_sensitive ( EN_N_estudiantes , 0 ) ;
8  gtk_widget_set_sensitive ( EN_institucion , 0 ) ;
9  gtk_widget_set_sensitive ( EN_escuela , 0 ) ;
10 gtk_widget_set_sensitive ( EN_programa , 0 ) ;
11 gtk_widget_set_sensitive ( EN_profesor , 0 ) ;
12 gtk_widget_set_sensitive ( EN_descripcion , 0 ) ;
13 gtk_widget_set_sensitive ( FR_prediccion , 0 ) ;
14 gtk_widget_set_sensitive
```

Código Preprocesado (Sin Pretty Print)

```
1  ( FR_real , 0 ) ;
2  gtk_widget_set_sensitive ( FR_grafico , 0 ) ;
3  gtk_widget_set_sensitive ( FR_preguntas , 0 ) ;
4  gtk_widget_set_sensitive ( FR_beamer , 0 ) ;
5  gtk_spin_button_set_value ( SP_resolucion , 10 ) ;
6  gtk_spin_button_set_value ( SP_color , 7 ) ;
7  gtk_spin_button_set_value ( SP_rotacion , 30 ) ;
8  gtk_toggle_button_set_active ( CK_smooth , TRUE ) ;
9  gtk_range_set_range ( GTK_RANGE ( SC_preguntas ) , (
       gdouble ) 0.0 , ( gdouble ) 1.0 ) ;
10 gtk_range_set_value ( GTK_RANGE ( SC_preguntas ) , (
       gdouble ) 0.0 ) ;
11 gtk_text_buffer_set_text ( buffer_TV_pregunta , "\0" ,
       - 1 ) ;
12 gtk_toggle_button_set_active ( TG_A , FALSE ) ;
13 gtk_toggle_button_set_active ( TG_B , FALSE ) ;
14 gtk_toggle_button_set_active
```

Código Preprocesado (Sin Pretty Print)

```
1  ( TG_C , FALSE ) ;
2  gtk_toggle_button_set_active ( TG_D , FALSE ) ;
3  gtk_toggle_button_set_active ( TG_E , FALSE ) ;
4  gtk_combo_box_set_active ( CB_ajuste , - 1 ) ;
5  gtk_toggle_button_set_active ( CK_no_actualiza , FALSE
       ) ;
6  gtk_toggle_button_set_active ( CK_excluir , FALSE ) ;
7  gtk_toggle_button_set_active ( CK_encoger , FALSE ) ;
8  gtk_toggle_button_set_active ( CK_verbatim , FALSE ) ;
9  gtk_toggle_button_set_active ( CK_header_encoger ,
       FALSE ) ;
10 gtk_toggle_button_set_active ( CK_header_verbatim ,
       FALSE ) ;
11 for ( i = 0 ; i < 5 ; i ++ )
12 {
13 gtk_toggle_button_set_active ( CK_slide [ i ] , FALSE )
       ;
14 gtk_toggle_button_set_active
```

Código Preprocesado (Sin Pretty Print)

```
1  ( CK_encoger_opcion [ i ] , FALSE ) ;
2  gtk_toggle_button_set_active ( CK_verbatim_opcion [ i ]
       , FALSE ) ;
3  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_encoger_opcion [ i ] ) , 0 ) ;
4  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_verbatim_opcion [ i ] ) , 0 ) ;
5  }
6  gtk_widget_set_sensitive ( BN_save , 0 ) ;
7  gtk_widget_set_sensitive ( BN_slides , 0 ) ;
8  gtk_widget_set_sensitive ( BN_print , 0 ) ;
9  gtk_widget_set_sensitive ( BN_undo , 1 ) ;
10 gdk_color_parse ( SECONDARY_AREA , & color ) ;
11 gtk_widget_modify_bg ( EB_ajustes , GTK_STATE_NORMAL ,
       & color ) ;
12 gtk_combo_box_set_active ( CB_estilo , parametros .
       Beamer_Estilo ) ;
13 gtk_combo_box_set_active ( CB_color , parametros .
       Beamer_Color ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( CB_font , parametros . Beamer_Font ) ;
2  gtk_combo_box_set_active ( CB_size , parametros .
       Beamer_Size ) ;
3  gtk_combo_box_set_active ( CB_aspecto , parametros .
       Beamer_Aspecto ) ;
4  gtk_toggle_button_set_active ( CK_general , FALSE ) ;
5  gtk_toggle_button_set_active ( CK_sin_banderas , FALSE
       ) ;
6  if ( Last > 1 )
7  gtk_widget_grab_focus ( GTK_WIDGET ( SP_examen ) ) ;
8  else
9  Cambio_Examen ( ) ;
10 }
11 void Cambio_Examen ( )
12 {
13 char examen [ 10 ] ;
14 char
```

Código Preprocesado (Sin Pretty Print)

```
1 hilera [ 200 ] ;
2 char Descripcion [ 2 *  201 ] = "*** Examen no est
      registrado ***" ;
3 char PG_command [ 4000 ] ;
4 long double media_prediccion , desviacion_prediccion ,
      alfa_prediccion , Rpb_prediccion ;
5 PGresult * res , * res_aux , * res_aux_2 ;
6 int i , j , k ;
7 k = ( int ) gtk_spin_button_get_value_as_int (
      SP_examen ) ;
8 sprintf ( examen , "%05d" , k ) ;
9 sprintf ( PG_command , "SELECT codigo_examen ,
      descripcion , pre_examen , profesor , EX_examenes.
      materia , institucion , escuela , programa , esquema ,
      descripcion_pre_examen , descripcion_esquema ,
      descripcion_materia , nombre , prediccion_media ,
      prediccion_desviacion , prediccion_alfa ,
      prediccion_Rpb , Year , month , day , ejecutado from
      EX_examenes , EX_pre_examenes , EX_esquemas ,
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EN_pre_examen , 1 ) ;
2  gtk_widget_set_sensitive ( EN_pre_examen_descripcion ,
       1 ) ;
3  gtk_widget_set_sensitive ( EN_esquema , 1 ) ;
4  gtk_widget_set_sensitive ( EN_esquema_descripcion , 1 )
       ;
5  gtk_widget_set_sensitive ( EN_materia , 1 ) ;
6  gtk_widget_set_sensitive ( EN_materia_descripcion , 1 )
       ;
7  gtk_widget_set_sensitive ( EN_fecha , 1 ) ;
8  gtk_widget_set_sensitive ( EN_N_preguntas , 1 ) ;
9  gtk_widget_set_sensitive ( EN_N_versiones , 1 ) ;
10 gtk_widget_set_sensitive ( EN_N_estudiantes , 1 ) ;
11 gtk_widget_set_sensitive ( EN_institucion , 1 ) ;
12 gtk_widget_set_sensitive ( EN_escuela , 1 ) ;
13 gtk_widget_set_sensitive ( EN_programa , 1 ) ;
14 gtk_widget_set_sensitive
```

Código Preprocesado (Sin Pretty Print)

```
1  ( EN_profesor , 1 ) ;
2  gtk_widget_set_sensitive ( EN_descripcion , 1 ) ;
3  gtk_widget_set_sensitive ( FR_prediccion , 1 ) ;
4  gtk_widget_set_sensitive ( FR_real , 1 ) ;
5  gtk_widget_set_sensitive ( FR_ajustes , 1 ) ;
6  gtk_widget_set_sensitive ( FR_grafico , 1 ) ;
7  gtk_widget_set_sensitive ( FR_preguntas , 1 ) ;
8  gtk_widget_set_sensitive ( FR_beamer , 1 ) ;
9  gtk_widget_set_sensitive ( BN_save , 1 ) ;
10 gtk_widget_set_sensitive ( BN_slides , 1 ) ;
11 gtk_widget_set_sensitive ( BN_print , 1 ) ;
12 sprintf ( PG_command , "SELECT DISTINCT version from
       EX_examenes_preguntas where examen = '%s' order by
       version" , examen ) ;
13 res_aux = PQEXEC ( DATABASE , PG_command ) ;
14 N_versiones
```

Código Preprocesado (Sin Pretty Print)

```
1  = PQntuples ( res_aux ) ;
2  PQclear ( res_aux ) ;
3  sprintf ( PG_command , "SELECT * from
       EX_examenes_preguntas where examen = '%s'" , examen
       ) ;
4  res_aux = PQEXEC ( DATABASE , PG_command ) ;
5  N_preguntas = PQntuples ( res_aux ) ;
6  N_preguntas = N_preguntas / N_versiones ;
7  PQclear ( res_aux ) ;
8  strcpy ( Descripcion , PQgetvalue ( res , 0 , 1 ) ) ;
9  gtk_entry_set_text ( GTK_ENTRY ( EN_pre_examen ) ,
       PQgetvalue ( res , 0 , 2 ) ) ;
10 gtk_entry_set_text ( GTK_ENTRY (
       EN_pre_examen_descripcion ) , PQgetvalue ( res , 0
       , 9 ) ) ;
11 gtk_entry_set_text ( GTK_ENTRY ( EN_esquema ) ,
       PQgetvalue ( res , 0 , 8 ) ) ;
12 gtk_entry_set_text ( GTK_ENTRY ( EN_esquema_descripcion
       ) , PQgetvalue ( res , 0 , 10 ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_ENTRY ( EN_materia ) , PQgetvalue ( res , 0 , 4 )
       ) ;
2  gtk_entry_set_text ( GTK_ENTRY ( EN_materia_descripcion
       ) , PQgetvalue ( res , 0 , 11 ) ) ;
3  gtk_entry_set_text ( GTK_ENTRY ( EN_institucion ) ,
     PQgetvalue ( res , 0 , 5 ) ) ;
4  gtk_entry_set_text ( GTK_ENTRY ( EN_escuela ) ,
     PQgetvalue ( res , 0 , 6 ) ) ;
5  gtk_entry_set_text ( GTK_ENTRY ( EN_programa ) ,
     PQgetvalue ( res , 0 , 7 ) ) ;
6  sprintf ( hilera , "%7d" , N_preguntas ) ;
7  gtk_entry_set_text ( GTK_ENTRY ( EN_N_preguntas ) ,
     hilera ) ;
8  sprintf ( hilera , "%7d" , N_versiones ) ;
9  gtk_entry_set_text ( GTK_ENTRY ( EN_N_versiones ) ,
     hilera ) ;
10 media_prediccion = atof ( PQgetvalue ( res , 0 , 13 ) )
       ;
11 sprintf ( hilera , "%8.3Lf" , media_prediccion ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera , "%8.3Lf" , desviacion_prediccion ) ;
2  gtk_entry_set_text ( GTK_ENTRY (
       EN_desviacion_prediccion ) , hilera ) ;
3  alfa_prediccion = atof ( PQgetvalue ( res , 0 , 15 ) )
       ;
4  sprintf ( hilera , "%7.4Lf" , alfa_prediccion ) ;
5  gtk_entry_set_text ( GTK_ENTRY ( EN_alfa_prediccion ) ,
        hilera ) ;
6  Rpb_prediccion = atof ( PQgetvalue ( res , 0 , 16 ) ) ;
7  sprintf ( hilera , "%7.4Lf" , Rpb_prediccion ) ;
8  gtk_entry_set_text ( GTK_ENTRY ( EN_Rpb_prediccion ) ,
       hilera ) ;
9  sprintf ( hilera , "%02d/%02d/%02d" , atoi ( PQgetvalue
        ( res , 0 , 19 ) ) , atoi ( PQgetvalue ( res , 0 ,
        18 ) ) , atoi ( PQgetvalue ( res , 0 , 17 ) ) ) ;
10 gtk_entry_set_text ( GTK_ENTRY ( EN_fecha ) , hilera )
       ;
11 preguntas = ( struct PREGUNTA * ) malloc ( ( sizeof (
       struct PREGUNTA ) * N_preguntas ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( examen ) ;
2  Carga_preguntas_examen ( ) ;
3  Actualiza_Porcentajes ( ) ;
4  Inicializa_Tabla_estadisticas ( ) ;
5  Calcula_estadisticas_examen ( ) ;
6  if ( * PQgetvalue ( res , 0 , 20 ) == 't' )
7  {
8  gtk_widget_set_sensitive ( FR_ajustes , 0 ) ;
9  gtk_widget_set_sensitive ( window1 , 0 ) ;
10 gtk_widget_show ( window2 ) ;
11 }
12 else
13 {
14 gtk_widget_set_sensitive
```

Código Preprocesado (Sin Pretty Print)

```
1  ( FR_ajustes , 1 ) ;
2  }
3  if ( N_estudiantes )
4  gtk_widget_grab_focus ( BN_print ) ;
5  else
6  gtk_widget_grab_focus ( BN_undo ) ;
7  }
8  else
9  gtk_widget_grab_focus ( GTK_WIDGET ( SP_examen ) ) ;
10 PQclear ( res ) ;
11 gtk_entry_set_text ( GTK_ENTRY ( EN_descripcion ) ,
       Descripcion ) ;
12 }
13 void Construye_versiones ( char * examen )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int i , j , k ;
3  char PG_command [ 3000 ] ;
4  PGresult * res , * res_aux ;
5  int respuesta_original ;
6  int opcion_1 , opcion_2 , opcion_3 , opcion_4 ,
       opcion_5 ;
7  sprintf ( PG_command , "SELECT DISTINCT version from
       EX_examenes_preguntas where examen = '%s' order by
       version" , examen ) ;
8  res = PQEXEC ( DATABASE , PG_command ) ;
9  N_versiones = PQntuples ( res ) ;
10 versiones = ( struct VERSION * ) malloc ( sizeof (
       struct VERSION ) * N_versiones ) ;
11 for ( i = 0 ; i < N_versiones ; i ++ )
12 {
13 strcpy ( versiones [ i ] . codigo , PQgetvalue ( res ,
       i , 0 ) ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( PG_command , "SELECT respuesta , opcion_1 , opcion_2 ,
       opcion_3 , opcion_4 , opcion_5 , codigo_pregunta from
       ex_examenes_preguntas , bd_texto_preguntas where
       examen = '%.5s' and version = '%.4s' and
       codigo_pregunta = codigo_unico_pregunta order by
       posicion" , examen , PQgetvalue ( res , i , 0 ) ) ;
2  res_aux = PQEXEC ( DATABASE , PG_command ) ;
3  N_preguntas = PQntuples ( res_aux ) ;
4  versiones [ i ] . preguntas = ( struct PREGUNTA_VERSION
       * ) malloc ( sizeof ( struct PREGUNTA_VERSION ) *
       ( N_preguntas ) ) ;
5  for ( j = 0 ; j < N_preguntas ; j ++ )
6  {
7  respuesta_original = * PQgetvalue ( res_aux , j , 0 ) -
       'A' ;
8  opcion_1 = atoi ( PQgetvalue ( res_aux , j , 1 ) ) ;
9  opcion_2 = atoi ( PQgetvalue ( res_aux , j , 2 ) ) ;
10 opcion_3 = atoi ( PQgetvalue ( res_aux , j , 3 ) ) ;
11 opcion_4 = atoi ( PQgetvalue ( res_aux , j , 4 ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( respuesta_original == opcion_2 ) versiones [ i ] .
       preguntas [ j ] . respuesta = 'B' ;
2  if ( respuesta_original == opcion_3 ) versiones [ i ] .
       preguntas [ j ] . respuesta = 'C' ;
3  if ( respuesta_original == opcion_4 ) versiones [ i ] .
       preguntas [ j ] . respuesta = 'D' ;
4  if ( respuesta_original == opcion_5 ) versiones [ i ] .
       preguntas [ j ] . respuesta = 'E' ;
5  versiones [ i ] . preguntas [ j ] . orden_version [ 0 ]
       = opcion_1 ;
6  versiones [ i ] . preguntas [ j ] . orden_version [ 1 ]
       = opcion_2 ;
7  versiones [ i ] . preguntas [ j ] . orden_version [ 2 ]
       = opcion_3 ;
8  versiones [ i ] . preguntas [ j ] . orden_version [ 3 ]
       = opcion_4 ;
9  versiones [ i ] . preguntas [ j ] . orden_version [ 4 ]
       = opcion_5 ;
10 strcpy ( versiones [ i ] . preguntas [ j ] . codigo ,
```

Código Preprocesado (Sin Pretty Print)

```
1  ( res ) ;
2  }
3  void Actualiza_Porcentajes ( )
4  {
5  char PG_command [ 2000 ] ;
6  PGresult * res , * res_aux ;
7  int i , j , k ;
8  int N_estudiantes ;
9  long double Porcentaje , Porcentaje_ajustado ;
10 char examen [ 10 ] ;
11 int N_correctas , N_ajustado , M_ajustado ;
12 k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
13 sprintf ( examen , "%05d" , k ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( PG_command , "SELECT estudiante , version , respuestas
       from EX_examenes_respuestas where examen = '%s'
       order by estudiante" , examen ) ;
2  res = PQEXEC ( DATABASE , PG_command ) ;
3  N_estudiantes = PQntuples ( res ) ;
4  for ( i = 0 ; i < N_estudiantes ; i ++ )
5  {
6  Calcula_Notas ( PQgetvalue ( res , i , 1 ) ,
7  PQgetvalue ( res , i , 2 ) ,
8  & N_correctas , & N_ajustado , & M_ajustado ) ;
9  Porcentaje = ( long double ) N_correctas / N_preguntas
       * 100.0 ;
10 if ( M_ajustado )
11 Porcentaje_ajustado = ( long double ) N_ajustado /
       M_ajustado * 100.0 ;
12 else
13 Porcentaje_ajustado = 0.0 ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( PG_command , "UPDATE ex_examenes_respuestas set
       correctas = %d, porcentaje = %Lf, nota_ajustada = %
       Lf, nota_final = %Lf where examen = '%s' and
       estudiante = %d" ,
2  N_correctas , Porcentaje , Porcentaje_ajustado ,
       Porcentaje_ajustado ,
3  examen , i + 1 ) ;
4  res_aux = PQEXEC ( DATABASE , PG_command ) ;
5  }
6  }
7  void Carga_preguntas_examen ( )
8  {
9  int i , j , k ;
10 char PG_command [ 3000 ] ;
11 PGresult * res , * res_aux ;
12 gchar * materia ;
13 char examen [ 10 ] ;
14 char
```

Código Preprocesado (Sin Pretty Print)

```
1 ejercicio_actual [ 10 ] ;
2 struct PREGUNTA temporal ;
3 unsigned int ajustes_Beamer ;
4 k = ( int ) gtk_spin_button_get_value_as_int (
      SP_examen ) ;
5 sprintf ( examen , "%05d" , k ) ;
6 materia = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_materia ) , 0 , -1 ) ;
7 sprintf ( PG_command , "SELECT tema, subtema, ejercicio
      , secuencia, codigo_pregunta, dificultad,
      BD_texto_preguntas.respuesta, texto_pregunta,
      nombre from EX_examenes_preguntas, BD_ejercicios,
      BD_texto_preguntas, BD_estadisticas_preguntas,
      BD_texto_ejercicios, BD_personas where examen = '%s
      ' and codigo_ejercicio = ejercicio and materia = '%
      s' and codigo_pregunta = pregunta and
      codigo_unico_pregunta = pregunta and
      consecutivo_texto = texto_ejercicio and
      codigo_persona = autor order by version, tema,
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ i ] . ejercicio , PQgetvalue ( res , i ,
       2 ) ) ;
2  preguntas [ i ] . secuencia = atoi ( PQgetvalue ( res ,
       i , 3 ) ) ;
3  strcpy ( preguntas [ i ] . pregunta , PQgetvalue ( res
       , i , 4 ) ) ;
4  preguntas [ i ] . previo = atof ( PQgetvalue ( res , i
       , 5 ) ) ;
5  preguntas [ i ] . correcta = * PQgetvalue ( res , i , 6
       ) ;
6  strncpy ( preguntas [ i ] . texto_pregunta , PQgetvalue
       ( res , i , 7 ) , 501 - 1 ) ;
7  preguntas [ i ] . texto_pregunta [ 501 - 1 ] = '\0' ;
8  strcpy ( preguntas [ i ] . autor , PQgetvalue ( res , i
       , 8 ) ) ;
9  preguntas [ i ] . desviacion = sqrt ( preguntas [ i ] .
       previo * ( 1.0 - preguntas [ i ] . previo ) ) ;
10 preguntas [ i ] . ajuste = 0 ;
11 preguntas [ i ] . revision_especial = 0 ;
```

Código Preprocesado (Sin Pretty Print)

```
1  [ i ] . encoger = 0 ;
2  preguntas [ i ] . verbatim = 0 ;
3  preguntas [ i ] . header_encoger = 0 ;
4  preguntas [ i ] . header_verbatim = 0 ;
5  for ( j = 0 ; j < 5 ; j ++ )
6  {
7  preguntas [ i ] . correctas_nuevas [ j ] = 0 ;
8  preguntas [ i ] . slide [ j ] = 0 ;
9  preguntas [ i ] . encoger_opcion [ j ] = 0 ;
10 preguntas [ i ] . verbatim_opcion [ j ] = 0 ;
11 }
12 sprintf ( PG_command , "SELECT * from
       EX_examenes_ajustes where examen = '%s' and
       codigo_pregunta = '%s'" , examen , PQgetvalue ( res
        , i , 4 ) ) ;
13 res_aux = PQEXEC ( DATABASE , PG_command ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( PQntuples ( res_aux ) )
2  {
3  preguntas [ i ] . ajuste = atoi ( PQgetvalue ( res_aux
       , 0 , 3 ) ) ;
4  preguntas [ i ] . revision_especial = preguntas [ i ] .
        ajuste & 0xFF ;
5  ajustes_Beamer = preguntas [ i ] . ajuste >> 8 ;
6  preguntas [ i ] . excluir = ajustes_Beamer & 0x01 ;
       ajustes_Beamer >>= 1 ;
7  preguntas [ i ] . encoger = ajustes_Beamer & 0x01 ;
       ajustes_Beamer >>= 1 ;
8  preguntas [ i ] . verbatim = ajustes_Beamer & 0x01 ;
       ajustes_Beamer >>= 1 ;
9  preguntas [ i ] . header_encoger = ajustes_Beamer & 0
       x01 ; ajustes_Beamer >>= 1 ;
10 preguntas [ i ] . header_verbatim = ajustes_Beamer & 0
       x01 ; ajustes_Beamer >>= 1 ;
11 for ( j = 0 ; j < 5 ; j ++ )
12 {
```

Código Preprocesado (Sin Pretty Print)

```
1  [ i ] . encoger_opcion [ j ] = ajustes_Beamer & 0x01 ;
       ajustes_Beamer >>= 1 ;
2  preguntas [ i ] . verbatim_opcion [ j ] =
       ajustes_Beamer & 0x01 ; ajustes_Beamer >>= 1 ;
3  }
4  preguntas [ i ] . correctas_nuevas [ 0 ] = atoi (
       PQgetvalue ( res_aux , 0 , 4 ) ) ;
5  preguntas [ i ] . correctas_nuevas [ 1 ] = atoi (
       PQgetvalue ( res_aux , 0 , 5 ) ) ;
6  preguntas [ i ] . correctas_nuevas [ 2 ] = atoi (
       PQgetvalue ( res_aux , 0 , 6 ) ) ;
7  preguntas [ i ] . correctas_nuevas [ 3 ] = atoi (
       PQgetvalue ( res_aux , 0 , 7 ) ) ;
8  preguntas [ i ] . correctas_nuevas [ 4 ] = atoi (
       PQgetvalue ( res_aux , 0 , 8 ) ) ;
9  preguntas [ i ] . actualizar = atoi ( PQgetvalue (
       res_aux , 0 , 9 ) ) ;
10 if ( preguntas [ i ] . revision_especial ) N_ajustes ++
       ;
```

Código Preprocesado (Sin Pretty Print)

```
1  = PQEXEC ( DATABASE , PG_command ) ;
2  preguntas [ i ] . orden_tema = atoi ( PQgetvalue (
       res_aux , 0 , 0 ) ) ;
3  PQclear ( res_aux ) ;
4  sprintf ( PG_command , "SELECT orden from BD_materias
       where codigo_materia = '%s' and codigo_tema = '%s'
       and codigo_subtema = '%s'" , materia , PQgetvalue (
        res , i , 0 ) , PQgetvalue ( res , i , 1 ) ) ;
5  res_aux = PQEXEC ( DATABASE , PG_command ) ;
6  preguntas [ i ] . orden_subtema = atoi ( PQgetvalue (
       res_aux , 0 , 0 ) ) ;
7  PQclear ( res_aux ) ;
8  }
9  PQclear ( res ) ;
10 for ( i = N_preguntas ; i >= 0 ; i -- )
11 for ( j = 0 ; j < ( i - 1 ) ; j ++ )
12 {
13 if ( ( preguntas [ j ] . orden_tema > preguntas [ j + 1
       ] . orden_tema ) ||
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ j ] . orden_tema == preguntas [ j + 1 ] .
       orden_tema ) &&
2  ( preguntas [ j ] . orden_subtema > preguntas [ j + 1 ]
       . orden_subtema ) ) ||
3  ( ( preguntas [ j ] . orden_tema == preguntas [ j + 1 ]
       . orden_tema ) &&
4  ( preguntas [ j ] . orden_subtema == preguntas [ j + 1
     ] . orden_subtema ) &&
5  strcmp ( preguntas [ j ] . ejercicio , preguntas [ j +
     1 ] . ejercicio ) > 0 ) )
6  {
7  temporal = preguntas [ j + 1 ] ;
8  preguntas [ j + 1 ] = preguntas [ j ] ;
9  preguntas [ j ] = temporal ;
10 }
11 }
12 g_free ( materia ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Inicializa_Tabla_estadisticas ( )
2  {
3  int i , j ;
4  char ejercicio_actual [ 10 ] ;
5  int inicio_actual ;
6  inicio_actual = 0 ;
7  strcpy ( ejercicio_actual , preguntas [ 0 ] . ejercicio
       ) ;
8  for ( i = 0 ; i < N_preguntas ; i ++ )
9  {
10 if ( strcmp ( ejercicio_actual , preguntas [ i ] .
       ejercicio ) != 0 )
11 {
12 strcpy ( ejercicio_actual , preguntas [ i ] . ejercicio
       ) ;
13 for ( j = inicio_actual ; j < i ; j ++ )
14 preguntas
```

Código Preprocesado (Sin Pretty Print)

```
1  [ j ] . grupo_final = i - 1 ;
2  inicio_actual = i ;
3  }
4  preguntas [ i ] . grupo_inicio = inicio_actual ;
5  preguntas [ i ] . buenos = 0 ;
6  preguntas [ i ] . malos = 0 ;
7  for ( j = 0 ; j < 5 ; j ++ )
8  {
9  preguntas [ i ] . acumulado_opciones [ j ] = 0 ;
10 preguntas [ i ] . suma_seleccion [ j ] = 0.0 ;
11 preguntas [ i ] . Rpb_opcion [ j ] = 0.0 ;
12 }
13 preguntas [ i ] . suma_buenos = 0.0 ;
14 preguntas
```

Código Preprocesado (Sin Pretty Print)

```
1  [ i ] . suma_malos = 0.0 ;
2  }
3  for ( i = inicio_actual ; i < N_preguntas ; i ++ )
4  preguntas [ i ] . grupo_final = N_preguntas - 1 ;
5  }
6  void Calcula_estadisticas_examen ( )
7  {
8  int i , j , k , N , version_actual ;
9  int buenas , malas ;
10 char PG_command [ 3000 ] ;
11 PGresult * res , * res_aux , * res_aux_2 ;
12 char examen [ 10 ] ;
13 char hilera [ 100 ] ;
14 long
```

Código Preprocesado (Sin Pretty Print)

```
1  double media , diferencia , varianza , desviacion ,
       varianza_pregunta , desviacion_pregunta ,
       suma_varianzas , alfa , Rpb ;
2  long double media_buenos , media_malos ;
3  long double media_sin , varianza_sin , alfa_sin ;
4  long double nota ;
5  char tema_actual [ CODIGO_TEMA_SIZE + 1 ] ;
6  char tema_subtema_actual [ CODIGO_SUBTEMA_SIZE + 1 ] ;
7  int buenas_tema , malas_tema , buenas_subtema ,
       malas_subtema ;
8  struct PREGUNTA temporal ;
9  int opcion_original [ 5 ] ;
10 k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
11 sprintf ( examen , "%05d" , k ) ;
12 for ( i = 0 ; i <  10 ; i ++ ) Frecuencias [ i ] = 0 ;
13 sprintf ( PG_command , "SELECT porcentaje , version ,
       respuestas , estudiante , nota_ajustada from
       EX_examenes_respuestas where examen = '%s' order by
```

Código Preprocesado (Sin Pretty Print)

```
= PQEXEC ( DATABASE , PG_command ) ;
N_estudiantes = PQntuples ( res ) ;
sprintf ( hilera , "%7d" , N_estudiantes ) ;
gtk_entry_set_text ( GTK_ENTRY ( EN_N_estudiantes ) ,
    hilera ) ;
if ( N_estudiantes <= 1 )
{
gtk_widget_set_sensitive ( FR_real , 0 ) ;
gtk_widget_set_sensitive ( FR_ajustes , 0 ) ;
gtk_widget_set_sensitive ( BN_save , 0 ) ;
gtk_widget_set_sensitive ( BN_slides , 0 ) ;
gtk_widget_set_sensitive ( BN_print , 0 ) ;
}
else
{
```

Código Preprocesado (Sin Pretty Print)

```
1
2  gtk_range_set_range ( GTK_RANGE ( SC_preguntas ) , (
       gdouble ) 1.0 , ( gdouble ) N_preguntas ) ;
3  gtk_range_set_value ( GTK_RANGE ( SC_preguntas ) , (
       gdouble ) 1.0 ) ;
4  media = 0.0 ;
5  Nota_minima = 100.0 ;
6  Nota_maxima = 0.0 ;
7  Nota_minima_ajustada = 100.0 ;
8  Nota_maxima_ajustada = 0.0 ;
9  for ( i = 0 ; i < N_estudiantes ; i ++ )
10 {
11 nota = atof ( PQgetvalue ( res , i , 0 ) ) ;
12 media += nota ;
13 if ( nota < Nota_minima ) Nota_minima = nota ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( nota > Nota_maxima ) Nota_maxima = nota ;
2  j = nota / ( 100.0 / ( long double ) 10 ) ;
3  Frecuencias [ j ] ++ ;
4  nota = atof ( PQgetvalue ( res , i , 4 ) ) ;
5  if ( nota < Nota_minima_ajustada ) Nota_minima_ajustada
       = nota ;
6  if ( nota > Nota_maxima_ajustada ) Nota_maxima_ajustada
       = nota ;
7  }
8  media /= N_estudiantes ;
9  sprintf ( hilera , "%8.3Lf" , media ) ;
10 gtk_entry_set_text ( GTK_ENTRY ( EN_media_real ) ,
       hilera ) ;
11 varianza = 0.0 ;
12 for ( i = 0 ; i < N_estudiantes ; i ++ )
13 {
14 diferencia
```

Código Preprocesado (Sin Pretty Print)

```
1  = media - atof ( PQgetvalue ( res , i , 0 ) ) ;
2  varianza += ( diferencia * diferencia ) ;
3  }
4  varianza /= N_estudiantes ;
5  desviacion = sqrt ( varianza ) ;
6  sprintf ( hilera , "%8.3Lf" , desviacion ) ;
7  gtk_entry_set_text ( GTK_ENTRY ( EN_desviacion_real ) ,
       hilera ) ;
8  for ( i = 0 ; i < N_estudiantes ; i ++ )
9  {
10 sprintf ( PG_command , "SELECT ejercicio , secuencia ,
       opcion_1 , opcion_2 , opcion_3 , opcion_4 , opcion_5
       from EX_examenes_preguntas where examen = '%s' and
       version = '%s' order by posicion" , examen ,
       PQgetvalue ( res , i , 1 ) ) ;
11 res_aux = PQEXEC ( DATABASE , PG_command ) ;
12 N = PQntuples ( res_aux ) ;
13 for ( version_actual = 0 ; ( version_actual <
       N_versiones ) && strcmp ( versiones [
```

Código Preprocesado (Sin Pretty Print)

```
1  = malas = 0 ;
2  for ( j = 0 ; j < N_preguntas ; j ++ )
3  {
4  opcion_original [ 0 ] = atoi ( PQgetvalue ( res_aux , j
       , 2 ) ) ;
5  opcion_original [ 1 ] = atoi ( PQgetvalue ( res_aux , j
       , 3 ) ) ;
6  opcion_original [ 2 ] = atoi ( PQgetvalue ( res_aux , j
       , 4 ) ) ;
7  opcion_original [ 3 ] = atoi ( PQgetvalue ( res_aux , j
       , 5 ) ) ;
8  opcion_original [ 4 ] = atoi ( PQgetvalue ( res_aux , j
       , 6 ) ) ;
9  if ( versiones [ version_actual ] . preguntas [ j ] .
      respuesta == PQgetvalue ( res , i , 2 ) [ j ] )
10 {
11 buenas ++ ;
12 }
13 else
```

Código Preprocesado (Sin Pretty Print)

```
1
2  malas ++ ;
3  }
4  Actualice_Estadisticas ( versiones [ version_actual ] .
       preguntas [ j ] . respuesta == PQgetvalue ( res ,
       i , 2 ) [ j ] ,
5  PQgetvalue ( res_aux , j , 0 ) , atoi ( PQgetvalue (
       res_aux , j , 1 ) ) , ( long double ) atof (
       PQgetvalue ( res , i , 0 ) ) ,
6  PQgetvalue ( res , i , 2 ) [ j ] , opcion_original ) ;
7  }
8  PQclear ( res_aux ) ;
9  }
10 Rpb = 0.0 ;
11 suma_varianzas = 0.0 ;
12 for ( i = 0 ; i < N_preguntas ; i ++ )
13 {
14 preguntas
```

Código Preprocesado (Sin Pretty Print)

```
1  [ i ] . porcentaje = ( long double ) preguntas [ i ] .
       buenos / N_estudiantes ;
2  varianza_pregunta = preguntas [ i ] . porcentaje * (
       1.0 - preguntas [ i ] . porcentaje ) ;
3  desviacion_pregunta = sqrt ( varianza_pregunta ) ;
4  suma_varianzas += varianza_pregunta ;
5  media_buenos = 0.0 ;
6  if ( preguntas [ i ] . buenos ) media_buenos =
       preguntas [ i ] . suma_buenos / preguntas [ i ] .
       buenos ;
7  media_malos = 0.0 ;
8  if ( preguntas [ i ] . malos ) media_malos = preguntas
       [ i ] . suma_malos / preguntas [ i ] . malos ;
9  preguntas [ i ] . Rpb = ( media_buenos - media_malos )
       * ( desviacion_pregunta / desviacion ) ;
10 Rpb += preguntas [ i ] . Rpb ;
11 for ( j = 0 ; j < 5 ; j ++ )
12 {
13 if ( ( preguntas [ i ] . acumulado_opciones [ j ] == 0
```

Código Preprocesado (Sin Pretty Print)

```
1
2  preguntas [ i ] . Rpb_opcion [ j ] = 0.0 ;
3  }
4  else
5  {
6  preguntas [ i ] . Rpb_opcion [ j ] = ( ( ( preguntas [
       i ] . suma_seleccion [ j ] / preguntas [ i ] .
       acumulado_opciones [ j ] ) -
7  ( ( preguntas [ i ] . suma_buenos + preguntas [ i ] .
       suma_malos - preguntas [ i ] . suma_seleccion [ j ]
        ) /
8  ( N_estudiantes - preguntas [ i ] . acumulado_opciones
       [ j ] ) ) ) /
9  desviacion ) *
10 sqrt ( ( ( long double ) preguntas [ i ] .
       acumulado_opciones [ j ] / N_estudiantes ) *
11 ( 1 - ( ( long double ) preguntas [ i ] .
       acumulado_opciones [ j ] / N_estudiantes ) ) ) ;
12 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  alfa = ( ( long double ) N_preguntas / ( long double )
       ( N_preguntas - 1 ) ) * ( 1.0 - ( suma_varianzas /
       varianza ) ) ;
3  sprintf ( hilera , "%8.3Lf" , alfa ) ;
4  gtk_entry_set_text ( GTK_ENTRY ( EN_alfa_real ) ,
       hilera ) ;
5  Rpb /= N_preguntas ;
6  sprintf ( hilera , "%8.3Lf" , Rpb ) ;
7  gtk_entry_set_text ( GTK_ENTRY ( EN_Rpb_real ) , hilera
       ) ;
8  for ( i = 0 ; i < N_preguntas ; i ++ )
9  {
10 media_sin = 0.0 ;
11 for ( j = 0 ; j < N_estudiantes ; j ++ )
12 {
13 sprintf ( PG_command , "SELECT version from
       EX_versiones where examen = '%s' and version <= '%s
       '" , examen , PQgetvalue ( res , j , 1 ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  = PQEXEC ( DATABASE , PG_command ) ;
2  version_actual = PQntuples ( res_aux_2 ) - 1 ;
3  PQclear ( res_aux_2 ) ;
4  N = 0 ;
5  for ( k = 0 ; k < N_preguntas ; k ++ ) N += ( versiones
       [ version_actual ] . preguntas [ k ] . respuesta
     == PQgetvalue ( res , j , 2 ) [ k ] ) ;
6  N -= ( versiones [ version_actual ] . preguntas [ i ] .
       respuesta == PQgetvalue ( res , j , 2 ) [ i ] ) ;
7  media_sin += ( ( long double ) N * 100.0 / (
     N_preguntas - 1 ) ) ;
8  }
9  media_sin /= N_estudiantes ;
10 varianza_sin = 0.0 ;
11 for ( j = 0 ; j < N_estudiantes ; j ++ )
12 {
13 sprintf ( PG_command , "SELECT version from
     EX_versiones where examen = '%s' and version <= '%s
     '" , examen , PQgetvalue ( res , j , 1 ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  = PQEXEC ( DATABASE , PG_command ) ;
2  version_actual = PQntuples ( res_aux_2 ) - 1 ;
3  PQclear ( res_aux_2 ) ;
4  N = 0 ;
5  for ( k = 0 ; k < N_preguntas ; k ++ ) N += ( versiones
       [ version_actual ] . preguntas [ k ] . respuesta
     == PQgetvalue ( res , j , 2 ) [ k ] ) ;
6  N -= ( versiones [ version_actual ] . preguntas [ i ] .
       respuesta == PQgetvalue ( res , j , 2 ) [ i ] ) ;
7  varianza_sin += ( ( media_sin - ( ( long double ) N *
     100.0 / ( N_preguntas - 1 ) ) ) * ( media_sin - ( (
      long double ) N * 100.0 / ( N_preguntas - 1 ) ) )
     ) ;
8  }
9  varianza_sin /= N_estudiantes ;
10 varianza_pregunta = ( ( ( long double ) preguntas [ i ]
       . buenos / N_estudiantes ) * ( ( long double )
     preguntas [ i ] . malos / N_estudiantes ) ) ;
11 preguntas [ i ] . alfa_sin = ( ( long double ) (
```

Código Preprocesado (Sin Pretty Print)

```
1  = 0 ;
2  N_subtemas = 0 ;
3  while ( i < N_preguntas )
4  {
5  strcpy ( resumen_tema [ N_temas ] . tema , preguntas [
       i ] . tema ) ;
6  strcpy ( resumen_tema [ N_temas ] . subtema ,
       CODIGO_VACIO ) ;
7  resumen_tema [ N_temas ] . buenos = 0 ;
8  resumen_tema [ N_temas ] . malos = 0 ;
9  while ( ( i < N_preguntas ) && ( strcmp ( resumen_tema
       [ N_temas ] . tema , preguntas [ i ] . tema ) == 0
       ) )
10 {
11 strcpy ( resumen_tema [ N_temas ] . subtema , preguntas
        [ i ] . subtema ) ;
12 strcpy ( resumen_tema_subtema [ N_subtemas ] . tema ,
       preguntas [ i ] . tema ) ;
13 strcpy ( resumen_tema_subtema [ N_subtemas ] . subtema
```

Código Preprocesado (Sin Pretty Print)

```
1  [ N_subtemas ] . buenos = 0 ;
2  resumen_tema_subtema [ N_subtemas ] . malos = 0 ;
3  while ( ( i < N_preguntas ) &&
4  ( strcmp ( resumen_tema [ N_temas ] . tema , preguntas
       [ i ] . tema ) == 0 ) &&
5  ( strcmp ( resumen_tema_subtema [ N_subtemas ] .
       subtema , preguntas [ i ] . subtema ) == 0 ) )
6  {
7  resumen_tema_subtema [ N_subtemas ] . buenos +=
       preguntas [ i ] . buenos ;
8  resumen_tema_subtema [ N_subtemas ] . malos +=
       preguntas [ i ] . malos ;
9  i ++ ;
10 }
11 resumen_tema [ N_temas ] . buenos +=
       resumen_tema_subtema [ N_subtemas ] . buenos ;
12 resumen_tema [ N_temas ] . malos +=
       resumen_tema_subtema [ N_subtemas ] . malos ;
13 N_subtemas ++ ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  N_temas ++ ;
3  }
4  for ( i = 0 ; i < N_temas ; i ++ ) resumen_tema [ i ] .
       porcentaje = ( long double ) resumen_tema [ i ] .
     buenos / ( resumen_tema [ i ] . buenos +
     resumen_tema [ i ] . malos ) ;
5  for ( i = N_temas ; i >= 0 ; i -- )
6  for ( j = 0 ; j < ( i - 1 ) ; j ++ )
7  {
8  if ( resumen_tema [ j ] . porcentaje > resumen_tema [ j
       + 1 ] . porcentaje )
9  {
10 temporal = resumen_tema [ j + 1 ] ;
11 resumen_tema [ j + 1 ] = resumen_tema [ j ] ;
12 resumen_tema [ j ] = temporal ;
13 }
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  for ( i = 0 ; i < N_subtemas ; i ++ )
       resumen_tema_subtema [ i ] . porcentaje = ( long
       double ) resumen_tema_subtema [ i ] . buenos / (
       resumen_tema_subtema [ i ] . buenos +
       resumen_tema_subtema [ i ] . malos ) ;
3  for ( i = N_subtemas ; i >= 0 ; i -- )
4  for ( j = 0 ; j < ( i - 1 ) ; j ++ )
5  {
6  if ( resumen_tema_subtema [ j ] . porcentaje >
       resumen_tema_subtema [ j + 1 ] . porcentaje )
7  {
8  temporal = resumen_tema_subtema [ j + 1 ] ;
9  resumen_tema_subtema [ j + 1 ] = resumen_tema_subtema [
         j ] ;
10 resumen_tema_subtema [ j ] = temporal ;
11 }
12 }
13 }
```

Código Preprocesado (Sin Pretty Print)

```
1  ( res ) ;
2  }
3  void Actualice_Estadisticas ( int buena , char *
       ejercicio , int secuencia , long double Nota , char
        respuesta , int opcion_original [ 5 ] )
4  {
5  int i , j ;
6  int found ;
7  for ( i = 0 ; ( i < N_preguntas ) && ! ( ( strcmp (
       ejercicio , preguntas [ i ] . ejercicio ) == 0 ) &&
        ( preguntas [ i ] . secuencia == secuencia ) ) ; i
        ++ ) ;
8  if ( buena )
9  {
10 preguntas [ i ] . buenos ++ ;
11 preguntas [ i ] . suma_buenos += Nota ;
12 }
13 else
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  preguntas [ i ] . malos ++ ;
3  preguntas [ i ] . suma_malos += Nota ;
4  }
5  if ( ( respuesta >= 'A' ) && ( respuesta <= 'E' ) )
6  {
7  preguntas [ i ] . acumulado_opciones [ opcion_original
       [ respuesta - 'A' ] ] ++ ;
8  preguntas [ i ] . suma_seleccion [ opcion_original [
       respuesta - 'A' ] ] += Nota ;
9  }
10 }
11 void Imprime_Reporte ( )
12 {
13 int k ;
14 FILE
```

Código Preprocesado (Sin Pretty Print)

```
1  * Archivo_Latex ;
2  long double media_real , desviacion_real ,
       media_prediccion , desviacion_prediccion , alfa ,
       Rpb , width ;
3  char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
4  gchar * materia , * materia_descripcion , * descripcion
       , * version , * institucion , * escuela , *
       programa , * fecha , * profesor ;
5  char Directorio [ 2000 ] ;
6  char comando [ 200 ] ;
7  char codigo [ 10 ] ;
8  DIR * pdir ;
9  gtk_widget_set_sensitive ( window1 , 0 ) ;
10 gtk_widget_show ( window4 ) ;
11 gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.0 ) ;
12 while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
13 Banderas [ 0 ] = Banderas [ 1 ] = Banderas [ 2 ] =
```

Código Preprocesado (Sin Pretty Print)

```
1  = ( int ) gtk_spin_button_get_value_as_int ( SP_examen
       ) ;
2  sprintf ( codigo , "%05d" , k ) ;
3  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
4  materia_descripcion = gtk_editable_get_chars (
       GTK_EDITABLE ( EN_materia_descripcion ) , 0 , -1 )
        ;
5  descripcion = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_descripcion ) , 0 , -1 ) ;
6  institucion = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_institucion ) , 0 , -1 ) ;
7  escuela = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_escuela ) , 0 , -1 ) ;
8  programa = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_programa ) , 0 , -1 ) ;
9  profesor = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_profesor ) , 0 , -1 ) ;
10 fecha = gtk_editable_get_chars ( GTK_EDITABLE (
```

Código Preprocesado (Sin Pretty Print)

```
1  = atof ( gtk_editable_get_chars ( GTK_EDITABLE (
       EN_desviacion_prediccion ) , 0 , -1 ) ) ;
2  alfa = atof ( gtk_editable_get_chars ( GTK_EDITABLE (
       EN_alfa_real ) , 0 , -1 ) ) ;
3  Rpb = atof ( gtk_editable_get_chars ( GTK_EDITABLE (
       EN_Rpb_real ) , 0 , -1 ) ) ;
4  Establece_Directorio ( Directorio , materia , fecha + 6
       , fecha + 3 , fecha ) ;
5  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.05 ) ;
6  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
7  Archivo_Latex = fopen ( "analisis.tex" , "w" ) ;
8  fprintf ( Archivo_Latex , "documentclass[9pt,journal,
       twoside, onecolumn]{EXAMINER}\n" ) ;
9  EX_latex_packages ( Archivo_Latex ) ;
10 fprintf ( Archivo_Latex , "\n%s\n\n" , parametros .
       Paquetes ) ;
11 fprintf ( Archivo_Latex , "graphicspath{{%s/}}\n\n" ,
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "fancyhead{}\n" ) ;
2  fprintf ( Archivo_Latex , "fancyhead[LO, RE]{%s}\n" ,
       fecha ) ;
3  fprintf ( Archivo_Latex , "fancyhead[LE,RO]{thepage}\n"
       ) ;
4  sprintf ( hilera_antes , "fancyhead[CE,CO]{textbf{%s -
       %s}}" , descripcion , materia_descripcion ) ;
5  hilera_LATEX ( hilera_antes , hilera_despues ) ;
6  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
7  fprintf ( Archivo_Latex , "fancyfoot{}\n" ) ;
8  sprintf ( hilera_antes , "fancyfoot[CE,CO]{%s}" ,
       institucion ) ;
9  hilera_LATEX ( hilera_antes , hilera_despues ) ;
10 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
11 sprintf ( hilera_antes , "fancyfoot[LE,LO]{textbf{%s}}"
       , escuela ) ;
12 hilera_LATEX ( hilera_antes , hilera_despues ) ;
13 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera_antes , "fancyfoot[RE,RO]{%s}" , programa ) ;
2  hilera_LATEX ( hilera_antes , hilera_despues ) ;
3  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
4  fprintf ( Archivo_Latex , "renewcommand{headrulewidth
       }{0.4pt}\n" ) ;
5  fprintf ( Archivo_Latex , "renewcommand{footrulewidth
       }{0.4pt}\n" ) ;
6  fprintf ( Archivo_Latex , "begin{document}\n" ) ;
7  fprintf ( Archivo_Latex , "\n\n" ) ;
8  sprintf ( hilera_antes , "title{An'{a}lisis de %s%s (%s
       )}" , descripcion , materia_descripcion , fecha ) ;
9  hilera_LATEX ( hilera_antes , hilera_despues ) ;
10 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
11 fprintf ( Archivo_Latex , "maketitle\n" ) ;
12 gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.125 ) ;
13 while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
14 Tabla_Datos_Generales
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , institucion , escuela , programa ,
       materia_descripcion , profesor , descripcion ,
       fecha , codigo , media_real , desviacion_real ,
       alfa , Rpb , 0 ) ;
2  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.15 ) ;
3  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
4  Analisis_General ( Archivo_Latex , alfa , Rpb , 0 ) ;
5  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.175 ) ;
6  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
7  width = media_real - Nota_minima ;
8  if ( ( Nota_maxima - media_real ) > width ) width =
       Nota_maxima - media_real ;
9  Prepara_Histograma_Notas ( ) ;
10 gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.22 ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( gtk_events_pending ( ) ) gtk_main_iteration ( ) ;
2  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
3  fprintf ( Archivo_Latex , "centering\n" ) ;
4  fprintf ( Archivo_Latex , "begin{minipage}[c]{0.49
       textwidth}\n" ) ;
5  fprintf ( Archivo_Latex , "includegraphics[scale=0.8]{
       EX4010-h.pdf}\n" ) ;
6  fprintf ( Archivo_Latex , "caption*{small {textbf{
       Histograma de Notas}}}\n" ) ;
7  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
8  fprintf ( Archivo_Latex , "begin{minipage}[c]{0.49
       textwidth}\n" ) ;
9  fprintf ( Archivo_Latex , "includegraphics[scale=0.8]{
       EX4010-n.pdf}\n" ) ;
10 fprintf ( Archivo_Latex , "caption*{small {textbf{
       Distribuci'{o}n Normal (real y predicci'{o}n)}}}\n"
        ) ;
11 fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
12 fprintf ( Archivo_Latex , "end{figure}\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( ) ;
2  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.35 ) ;
3  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
4  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
5  fprintf ( Archivo_Latex , "centering\n" ) ;
6  fprintf ( Archivo_Latex , "includegraphics[scale=1.0]{
       EX4010p.pdf}\n" ) ;
7  hilera_LATEX ( "caption*{textbf{Cruce entre Coeficiente
       de Discriminaci n ($r_{pb}$) y Dificultad ($p$)}}
       " , hilera_despues ) ;
8  fprintf ( Archivo_Latex , "%s\n\n" , hilera_despues ) ;
9  fprintf ( Archivo_Latex , "end{figure}\n\n" ) ;
10 fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
11 fprintf ( Archivo_Latex , "centering\n" ) ;
12 fprintf ( Archivo_Latex , "includegraphics[scale=1.0]{
       EX4010c.pdf}\n" ) ;
13 hilera_LATEX ( "caption*{textbf{L neas de Contorno del
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "%s\n\n" , hilera_despues ) ;
2  fprintf ( Archivo_Latex , "end{figure}\n\n" ) ;
3  fprintf ( Archivo_Latex , "clearpage\n" ) ;
4  if ( N_temas > 1 ) Prepara_Grafico_Pastel (
       Archivo_Latex ) ;
5  Prepara_Histograma_Temas ( ) ;
6  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
7  fprintf ( Archivo_Latex , "centering\n" ) ;
8  fprintf ( Archivo_Latex , "includegraphics[scale=1.0,
       angle=-90]{EX4010-t.pdf}\n" ) ;
9  fprintf ( Archivo_Latex , "caption*{small {textbf{An'{a
       }lisis por Temas (ordenado por rendimiento)}}}\n" )
        ;
10 fprintf ( Archivo_Latex , "end{figure}\n" ) ;
11 gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.4 ) ;
12 while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
        ;
13 Prepara_Histograma_Subtemas ( ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{figure}[H]\n" ) ;
2  fprintf ( Archivo_Latex , "centering\n" ) ;
3  fprintf ( Archivo_Latex , "includegraphics[scale=1.0,
       angle=-90]{EX4010-s.pdf}\n" ) ;
4  fprintf ( Archivo_Latex , "caption*{small {textbf{An'{a
       }lisis por Subtemas (ordenado por rendimiento)}}}\n
       " ) ;
5  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
6  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.45 ) ;
7  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
8  Asigna_Banderas ( ) ;
9  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.5 ) ;
10 while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
11 fprintf ( Archivo_Latex , "twocolumn\n" ) ;
12 fprintf ( Archivo_Latex , "clearpage\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , PB_analisis , 0.5 , 0.29 ) ;
2  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.8 ) ;
3  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
4  fprintf ( Archivo_Latex , "onecolumn\n" ) ;
5  fprintf ( Archivo_Latex , "clearpage\n" ) ;
6  fprintf ( Archivo_Latex , "SetWatermarkText{}\n" ) ;
7  Resumen_de_Banderas ( Archivo_Latex , 0 ) ;
8  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.82 ) ;
9  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
10 fprintf ( Archivo_Latex , "clearpage\n" ) ;
11 fprintf ( Archivo_Latex , "SetWatermarkText{
       Confidencial}\n" ) ;
12 Lista_de_Notas ( Archivo_Latex ) ;
13 fprintf ( Archivo_Latex , "end{document}\n" ) ;
14 fclose
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex ) ;
2  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.85 ) ;
3  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
4  latex_2_pdf ( & parametros , Directorio , parametros .
       ruta_latex , "analisis" , 1 , PB_analisis , 0.85 ,
       0.1 , NULL , NULL ) ;
5  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.95 ) ;
6  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
7  system ( "rm EX4010-h.pdf" ) ;
8  system ( "rm EX4010-n.pdf" ) ;
9  system ( "rm EX4010-t.pdf" ) ;
10 system ( "rm EX4010-s.pdf" ) ;
11 system ( "rm EX4010p.pdf" ) ;
12 system ( "rm EX4010c.pdf" ) ;
13 system ( "rm EX4010.dat" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( materia ) ;
2  g_free ( materia_descripcion ) ;
3  g_free ( descripcion ) ;
4  g_free ( institucion ) ;
5  g_free ( escuela ) ;
6  g_free ( programa ) ;
7  g_free ( fecha ) ;
8  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 1.0 ) ;
9  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
10 gtk_widget_hide ( window4 ) ;
11 gtk_widget_set_sensitive ( window1 , 1 ) ;
12 }
13 void Prepara_Grafico_Pastel ( FILE * Archivo_Latex )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int Empates ;
3  Crea_archivo_datos_pastel ( & Empates ) ;
4  fprintf ( Archivo_Latex , "DTLloaddb{datosesquema}{
       EX4010.dat}\n" ) ;
5  fprintf ( Archivo_Latex , "begin{figure}[htpb]\n" ) ;
6  fprintf ( Archivo_Latex , "centering\n" ) ;
7  fprintf ( Archivo_Latex , "setlength{DTLpieoutlinewidth
       }{1pt}\n" ) ;
8  colores_pastel ( Archivo_Latex ) ;
9  fprintf ( Archivo_Latex , "renewcommand*{
       DTLdisplayinnerlabel}[1]{textsf{#1}}\n" ) ;
10 fprintf ( Archivo_Latex , "renewcommand*{
       DTLdisplayouterlabel}[1]{textsf{#1}}\n" ) ;
11 fprintf ( Archivo_Latex , "DTLpiechart{variable=
       quantity,%%\n" ) ;
12 fprintf ( Archivo_Latex , "radius=4.1cm,%%\n" ) ;
13 fprintf ( Archivo_Latex , "outerratio=1.05,%%\n" ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Empates < 18 )
2  if ( Empates == 1 )
3  fprintf ( Archivo_Latex , "cutaway={1},%%\n" ) ;
4  else
5  fprintf ( Archivo_Latex , "cutaway={1-%d},%%\n" ,
       Empates ) ;
6  fprintf ( Archivo_Latex , "innerlabel={},%%\n" ) ;
7  fprintf ( Archivo_Latex , "outerlabel={DTLpievariable
       }}{datosesquema}{%%\n" ) ;
8  fprintf ( Archivo_Latex , "name=Name,quantity=Quantity
       }\n" ) ;
9  fprintf ( Archivo_Latex , "begin{tabular}[b]{ll}\n" ) ;
10 fprintf ( Archivo_Latex , "DTLforeach{datosesquema}{
       name=Name,quantity=Quantity}{DTLiffirstrow{}{}%%\n"
        ) ;
11 fprintf ( Archivo_Latex , "
       DTLdocurrentpiesegmentcolorrule{10pt}{10pt}&\n" ) ;
12 fprintf ( Archivo_Latex , "textsf{name}\n" ) ;
13 fprintf ( Archivo_Latex , "}\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "end{tabular}\n" ) ;
2  fprintf ( Archivo_Latex , "caption*{small {textbf{
       Material Evaluado en el Examen}}}\n" ) ;
3  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
4  }
5  void Prepara_Grafico_Pastel_Beamer ( FILE *
       Archivo_Latex )
6  {
7  int Empates ;
8  int aspecto ;
9  aspecto = gtk_combo_box_get_active ( CB_aspecto ) ;
10 Crea_archivo_datos_pastel ( & Empates ) ;
11 fprintf ( Archivo_Latex , "DTLloaddb{datosesquema}{
       EX4010.dat}\n" ) ;
12 fprintf ( Archivo_Latex , "begin{columns}\n" ) ;
13 fprintf ( Archivo_Latex , "begin{column}{0.5textwidth}\
       n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{figure}[H]\n" ) ;
2  fprintf ( Archivo_Latex , "centering\n" ) ;
3  fprintf ( Archivo_Latex , "setlength{DTLpieoutlinewidth
       }{1pt}\n" ) ;
4  colores_pastel ( Archivo_Latex ) ;
5  fprintf ( Archivo_Latex , "renewcommand*{
       DTLdisplayinnerlabel}[1]{textsf{#1}}\n" ) ;
6  fprintf ( Archivo_Latex , "renewcommand*{
       DTLdisplayouterlabel}[1]{textsf{#1}}\n" ) ;
7  fprintf ( Archivo_Latex , "DTLpiechart{variable=
       quantity,%%\n" ) ;
8  if ( ( aspecto == 1 ) || ( aspecto == 2 ) )
9  fprintf ( Archivo_Latex , "radius=3.0cm,%%\n" ) ;
10 else
11 fprintf ( Archivo_Latex , "radius=2.25cm,%%\n" ) ;
12 fprintf ( Archivo_Latex , "outerratio=1.05,%%\n" ) ;
13 if ( Empates <  18 )
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Empates == 1 )
2  fprintf ( Archivo_Latex , "cutaway={1},%%\n" ) ;
3  else
4  fprintf ( Archivo_Latex , "cutaway={1-%d},%%\n" ,
       Empates ) ;
5  fprintf ( Archivo_Latex , "innerlabel={},%%\n" ) ;
6  fprintf ( Archivo_Latex , "outerlabel={DTLpievariable
       }}{datosesquema}{%%\n" ) ;
7  fprintf ( Archivo_Latex , "name=Name,quantity=Quantity
       }\n" ) ;
8  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
9  fprintf ( Archivo_Latex , "end{column}\n" ) ;
10 fprintf ( Archivo_Latex , "begin{column}{0.5textwidth}\
       n" ) ;
11 fprintf ( Archivo_Latex , "begin{figure}[htpb]\n" ) ;
12 fprintf ( Archivo_Latex , "centering\n" ) ;
13 colores_pastel ( Archivo_Latex ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{tabular}[b]{ll}\n" ) ;
2  fprintf ( Archivo_Latex , "DTLforeach{datosesquema}{
       name=Name, quantity=Quantity}{DTLiffirstrow{}{}%%\n"
        ) ;
3  fprintf ( Archivo_Latex , "
       DTLdocurrentpiesegmentcolorrule{10pt}{10pt}&\n" ) ;
4  fprintf ( Archivo_Latex , "textsf{name}\n" ) ;
5  fprintf ( Archivo_Latex , "}\n" ) ;
6  fprintf ( Archivo_Latex , "end{tabular}\n" ) ;
7  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
8  fprintf ( Archivo_Latex , "end{column}\n" ) ;
9  fprintf ( Archivo_Latex , "end{columns}\n" ) ;
10 }
11 void Crea_archivo_datos_pastel ( int * Empates )
12 {
13 gchar * materia ;
14 FILE
```

Código Preprocesado (Sin Pretty Print)

```
1  * Archivo_Datos ;
2  int i , j , N , N_otros ;
3  char hilera_antes [ 2000 ] ;
4  char hilera_despues [ 2000 ] ;
5  struct {
6  char tema [ CODIGO_TEMA_SIZE + 1 ] ;
7  int cantidad ;
8  } temporal , tabla [ N_temas ] ;
9  char PG_command [ 2000 ] ;
10 PGresult * res_tema ;
11 char Descripcion [ 300 ] ;
12 int MAX_DESCRIPCION ;
13 int aspecto ;
14 aspecto
```

Código Preprocesado (Sin Pretty Print)

```
1  = gtk_combo_box_get_active ( CB_aspecto ) ;
2  if ( ( aspecto == 1 ) || ( aspecto == 2 ) )
3  MAX_DESCRIPCION = 43 ;
4  else
5  MAX_DESCRIPCION = 35 ;
6  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
7  Archivo_Datos = fopen ( "EX4010.dat" , "w" ) ;
8  fprintf ( Archivo_Datos , "Name, Quantity\n" ) ;
9  for ( i = 0 ; i < N_temas ; i ++ )
10 {
11 strcpy ( tabla [ i ] . tema , resumen_tema [ i ] . tema
       ) ;
12 tabla [ i ] . cantidad = ( resumen_tema [ i ] . buenos
     + resumen_tema [ i ] . malos ) / N_estudiantes ;
13 }
14 for
```

Código Preprocesado (Sin Pretty Print)

```
1  ( i = N_temas ; i >= 0 ; i -- )
2  for ( j = 0 ; j < ( i - 1 ) ; j ++ )
3  {
4  if ( tabla [ j ] . cantidad < tabla [ j + 1 ] .
       cantidad )
5  {
6  temporal = tabla [ j + 1 ] ;
7  tabla [ j + 1 ] = tabla [ j ] ;
8  tabla [ j ] = temporal ;
9  }
10 }
11 N = N_temas ;
12 if ( N >  18 ) N =  18 - 1 ;
13 for ( i = 0 ; i < N ; i ++ )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  sprintf ( PG_command , "SELECT descripcion_materia from
       bd_materias where codigo_materia = '%s' and
     codigo_tema = '%s' and codigo_subtema = '
     '" ,
3  materia , tabla [ i ] . tema ) ;
4  res_tema = PQEXEC ( DATABASE , PG_command ) ;
5  strcpy ( Descripcion , PQgetvalue ( res_tema , 0 , 0 )
       ) ;
6  if ( strlen ( Descripcion ) > MAX_DESCRIPCION )
7  {
8  Descripcion [ MAX_DESCRIPCION - 3 ] = '.' ;
9  Descripcion [ MAX_DESCRIPCION - 2 ] = '.' ;
10 Descripcion [ MAX_DESCRIPCION - 1 ] = '.' ;
11 Descripcion [ MAX_DESCRIPCION ] = '\0' ;
12 }
13 sprintf ( hilera_antes , "\"textbf{%s}\", %d" ,
     Descripcion , tabla [ i ] . cantidad ) ;
14 hilera_LATEX
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera_antes , hilera_despues ) ;
2  fprintf ( Archivo_Datos , "%s\n" , hilera_despues ) ;
3  }
4  N_otros = 0 ;
5  for ( i = N ; i < N_temas ; i ++ ) N_otros += tabla [ i
       ] . cantidad ;
6  if ( N_otros )
7  fprintf ( Archivo_Datos , "\"textbf{Otros} (%d temas)
       \", %d\n" , N_temas - N , N_otros ) ;
8  fclose ( Archivo_Datos ) ;
9  i = 1 ;
10 while ( ( i < N_temas ) && tabla [ 0 ] . cantidad ==
       tabla [ i ++ ] . cantidad ) ;
11 * Empates = i - 1 ;
12 g_free ( materia ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  colores_pastel ( FILE * Archivo_Latex )
2  {
3  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{1}{
       green}\n" ) ;
4  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{2}{
       blue}\n" ) ;
5  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{3}{red
       }\n" ) ;
6  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{4}{
       yellow}\n" ) ;
7  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{5}{
       magenta}\n" ) ;
8  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{6}{
       cyan}\n" ) ;
9  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{7}{
       orange}\n" ) ;
10 fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{8}{
       violet}\n" ) ;
11 fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{9}{
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "DTLsetpiesegmentcolor{12}{pink}\n" )
       ;
2  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{13}{
       brown}\n" ) ;
3  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{14}{
       purple}\n" ) ;
4  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{15}{
       darkgray}\n" ) ;
5  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{16}{
       olive}\n" ) ;
6  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{17}{
       black}\n" ) ;
7  fprintf ( Archivo_Latex , "DTLsetpiesegmentcolor{18}{
       lightgray}\n" ) ;
8  }
9  void Establece_Directorio ( char * Directorio , gchar *
       materia , char * year , char * month , char * day
       )
10 {
```

Código Preprocesado (Sin Pretty Print)

```
1  Directorio_materia_fecha [ 600 ] ;
2  int i , n ;
3  sprintf ( Directorio_materia , "%s/%s" , parametros .
       ruta_examenes , materia ) ;
4  n = strlen ( Directorio_materia ) ;
5  for ( i = n - 1 ; Directorio_materia [ i ] == ' ' ; i
       -- ) ;
6  Directorio_materia [ i + 1 ] = '\0' ;
7  sprintf ( Directorio , "%s/%s-%.2s-%.2s" ,
       Directorio_materia , year , month , day ) ;
8  pdir = opendir ( Directorio_materia ) ;
9  if ( ! pdir )
10 {
11 mkdir ( Directorio_materia , S_IRWXU | S_IRWXG |
       S_IROTH | S_IXOTH ) ;
12 }
13 else
14 closedir
```

Código Preprocesado (Sin Pretty Print)

```
1  ( pdir ) ;
2  pdir = opendir ( Directorio ) ;
3  if ( ! pdir )
4  {
5  mkdir ( Directorio , S_IRWXU | S_IRWXG | S_IROTH |
       S_IXOTH ) ;
6  }
7  else
8  closedir ( pdir ) ;
9  }
10 void Quita_espacios ( char * hilera )
11 {
12 int i , j ;
13 i = j = 0 ;
14 while
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera [ i ] )
2  {
3  if ( hilera [ i ] != ' ' ) hilera [ j ++ ] = hilera [ i
       ] ;
4  i ++ ;
5  }
6  hilera [ j ] = '\0' ;
7  }
8  void Genera_Beamer ( )
9  {
10 int k ;
11 char codigo [ 10 ] ;
12 gchar * size , * font , * color , * estilo , * materia
       , * materia_descripcion , * descripcion , * version
       , * institucion , * escuela , * programa , * fecha
       , * profesor ;
13 long double media_real , desviacion_real ,
       media_prediccion , desviacion_prediccion , alfa ,
       Rpb , width ;
```

Código Preprocesado (Sin Pretty Print)

```
1  * Archivo_Latex ;
2  char Directorio [ 3000 ] ;
3  char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
4  int aspecto ;
5  int resultado_OK ;
6  gtk_widget_set_sensitive ( window1 , 0 ) ;
7  gtk_widget_show ( window3 ) ;
8  Update_PB ( PB_beamer , 0.0 ) ;
9  Banderas [ 0 ] = Banderas [ 1 ] = Banderas [ 2 ] =
       Banderas [ 3 ] = Banderas [ 4 ] = Banderas [ 5 ] =
       0 ;
10 k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
11 sprintf ( codigo , "%05d" , k ) ;
12 materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
13 materia_descripcion = gtk_editable_get_chars (
       GTK_EDITABLE ( EN_materia_descripcion ) , 0 , -1 )
        ;
```

Código Preprocesado (Sin Pretty Print)

```
1 = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_descripcion ) , 0 , -1 ) ;
2 institucion = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_institucion ) , 0 , -1 ) ;
3 escuela = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_escuela ) , 0 , -1 ) ;
4 programa = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_programa ) , 0 , -1 ) ;
5 profesor = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_profesor ) , 0 , -1 ) ;
6 fecha = gtk_editable_get_chars ( GTK_EDITABLE (
      EN_fecha ) , 0 , -1 ) ;
7 media_real = atof ( gtk_editable_get_chars (
      GTK_EDITABLE ( EN_media_real ) , 0 , -1 ) ) ;
8 desviacion_real = atof ( gtk_editable_get_chars (
      GTK_EDITABLE ( EN_desviacion_real ) , 0 , -1 ) ) ;
9 media_prediccion = atof ( gtk_editable_get_chars (
      GTK_EDITABLE ( EN_media_prediccion ) , 0 , -1 ) )
      ;
```

Código Preprocesado (Sin Pretty Print)

```
1  = gtk_combo_box_get_active_text ( CB_color ) ;
2  font = gtk_combo_box_get_active_text ( CB_font ) ;
3  size = gtk_combo_box_get_active_text ( CB_size ) ;
4  aspecto = gtk_combo_box_get_active ( CB_aspecto ) ;
5  Establece_Directorio ( Directorio , materia , fecha + 6
       , fecha + 3 , fecha ) ;
6  Update_PB ( PB_beamer , 0.05 ) ;
7  Archivo_Latex = fopen ( "analisis-beamer.tex" , "w" ) ;
8  Beamer_Preamble ( Archivo_Latex , aspecto , size ,
       estilo , color , font , materia_descripcion ,
       descripcion , profesor , programa , escuela ,
       institucion , fecha ) ;
9  Update_PB ( PB_beamer , 0.08 ) ;
10 Beamer_Cover ( Archivo_Latex ) ;
11 Update_PB ( PB_beamer , 0.09 ) ;
12 Beamer_TOC ( Archivo_Latex ) ;
13 Update_PB ( PB_beamer , 0.10 ) ;
14 Beamer_Datos_Generales
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , institucion , escuela , programa ,
       materia_descripcion , profesor , descripcion ,
       fecha , codigo , media_real , desviacion_real ,
       alfa , Rpb ) ;
2  Update_PB ( PB_beamer , 0.12 ) ;
3  if ( N_temas > 1 ) Beamer_Grafico_Pastel (
       Archivo_Latex ) ;
4  Update_PB ( PB_beamer , 0.15 ) ;
5  Beamer_Histograma_Notas ( Archivo_Latex , media_real ,
       desviacion_real , media_prediccion ,
       desviacion_prediccion ) ;
6  Update_PB ( PB_beamer , 0.17 ) ;
7  Beamer_Dificultad_vs_Discriminacion ( Archivo_Latex ) ;
8  Update_PB ( PB_beamer , 0.19 ) ;
9  Beamer_Histograma_Temas ( Archivo_Latex ) ;
10 Update_PB ( PB_beamer , 0.21 ) ;
11 Beamer_Preguntas ( Archivo_Latex , PB_beamer , 0.21 ,
       0.33 ) ;
12 Beamer_Gracias ( Archivo_Latex ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex ) ;
2  Update_PB ( PB_beamer , 0.55 ) ;
3  resultado_OK = latex_2_pdf ( & parametros , Directorio
       , parametros . ruta_latex , "analisis-beamer" , 1 ,
        PB_beamer , 0.55 , 0.43 , NULL , NULL ) ;
4  system ( "rm EX4010-h.pdf" ) ;
5  system ( "rm EX4010-n.pdf" ) ;
6  system ( "rm EX4010-t.pdf" ) ;
7  system ( "rm EX4010p.pdf" ) ;
8  system ( "rm EX4010c.pdf" ) ;
9  system ( "rm EX4010.dat" ) ;
10 system ( "rm analisis-beamer.*" ) ;
11 Update_PB ( PB_beamer , 0.99 ) ;
12 g_free ( estilo ) ;
13 g_free ( color ) ;
14 g_free
```

Código Preprocesado (Sin Pretty Print)

```
1  ( font ) ;
2  g_free ( size ) ;
3  g_free ( materia ) ;
4  g_free ( materia_descripcion ) ;
5  g_free ( descripcion ) ;
6  g_free ( institucion ) ;
7  g_free ( escuela ) ;
8  g_free ( programa ) ;
9  g_free ( fecha ) ;
10 Update_PB ( PB_beamer , 1.0 ) ;
11 gtk_widget_hide ( window3 ) ;
12 gtk_widget_set_sensitive ( window1 , 1 ) ;
13 if ( ! resultado_OK )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  Beamer_Failure ( ) ;
3  }
4  }
5  void Beamer_Preamble ( FILE * Archivo_Latex , int
       aspecto , gchar * size , gchar * estilo , gchar *
       color , gchar * font , gchar * materia_descripcion
       , gchar * descripcion , gchar * profesor , gchar *
       programa , gchar * escuela , gchar * institucion ,
       gchar * fecha )
6  {
7  char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
8  fprintf ( Archivo_Latex , "documentclass[aspectratio=%s
       ,%s]{beamer}\n" , Beamer_aspectratio [ aspecto ] ,
       size ) ;
9  fprintf ( Archivo_Latex , "def BEAMER {}\n" ) ;
10 EX_latex_packages ( Archivo_Latex ) ;
11 fprintf ( Archivo_Latex , "\n%s\n\n" , parametros .
       Paquetes ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "graphicspath{{%s/}}\n\n" ,
       parametros . ruta_figuras ) ;
2  fprintf ( Archivo_Latex , "usetheme{%s}\n" , estilo ) ;
3  fprintf ( Archivo_Latex , "usecolortheme{%s}\n" , color
       ) ;
4  fprintf ( Archivo_Latex , "usefonttheme{%s}\n" , font )
       ;
5  fprintf ( Archivo_Latex , "useoutertheme{shadow}\n" ) ;
6  fprintf ( Archivo_Latex , "setbeamertemplate{navigation
       symbols}{}\n" ) ;
7  fprintf ( Archivo_Latex , "setbeamertemplate{caption}[
       numbered]\n" ) ;
8  fprintf ( Archivo_Latex , "captionsetup{labelsep =
       colon , figureposition = bottom}\n" ) ;
9  fprintf ( Archivo_Latex , "setbeamercolor{caption name
       }{fg=black}\n" ) ;
10 fprintf ( Archivo_Latex , "setbeamertemplate{headline
       }{}\n" ) ;
11 fprintf ( Archivo_Latex , "definecolor{DoradoPalido}{
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "SetWatermarkText{}\n" ) ;
2  sprintf ( hilera_antes , "title{%s}" ,
       materia_descripcion ) ;
3  hilera_LATEX ( hilera_antes , hilera_despues ) ;
4  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
5  sprintf ( hilera_antes , "subtitle{An lisis de %s}" ,
       descripcion ) ;
6  hilera_LATEX ( hilera_antes , hilera_despues ) ;
7  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
8  sprintf ( hilera_antes , "author[%s]{Prof. %s}" ,
       profesor , profesor ) ;
9  hilera_LATEX ( hilera_antes , hilera_despues ) ;
10 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
11 fprintf ( Archivo_Latex , "institute{\n" ) ;
12 sprintf ( hilera_antes , "%s" , programa ) ;
13 hilera_LATEX ( hilera_antes , hilera_despues ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "%s\n" , hilera_despues ) ;
2  sprintf ( hilera_antes , "%s" , escuela ) ;
3  hilera_LATEX ( hilera_antes , hilera_despues ) ;
4  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
5  sprintf ( hilera_antes , "%s" , institucion ) ;
6  hilera_LATEX ( hilera_antes , hilera_despues ) ;
7  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
8  fprintf ( Archivo_Latex , "}\n" ) ;
9  fprintf ( Archivo_Latex , "date{%s}\n" , fecha ) ;
10 fprintf ( Archivo_Latex , "begin{document}\n\n" ) ;
11 }
12 void Beamer_Cover ( FILE * Archivo_Latex )
13 {
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{frame}[plain]\n" ) ;
2  fprintf ( Archivo_Latex , "titlepage\n" ) ;
3  fprintf ( Archivo_Latex , "begin{figure}\n" ) ;
4  fprintf ( Archivo_Latex , "includegraphics[height=1.3cm
       ]{.imagenes/EX.png}\n" ) ;
5  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
6  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
7  }
8  void Beamer_TOC ( FILE * Archivo_Latex )
9  {
10 fprintf ( Archivo_Latex , "begin{frame}[plain]{
       Contenido}\n" ) ;
11 fprintf ( Archivo_Latex , "tableofcontents\n" ) ;
12 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1 Beamer_Datos_Generales ( FILE * Archivo_Latex , gchar *
      institucion , gchar * escuela , gchar * programa ,
      gchar * materia_descripcion , gchar * profesor ,
     gchar * descripcion , gchar * fecha , char * codigo
      , long double media_real , long double
     desviacion_real , long double alfa , long double
     Rpb )
2 {
3 fprintf ( Archivo_Latex , "section{Estad'isticas B'
     asicas}\n" ) ;
4 fprintf ( Archivo_Latex , "begin{frame}{Datos Generales
     }\n" ) ;
5 Tabla_Datos_Generales ( Archivo_Latex , institucion ,
     escuela , programa , materia_descripcion , profesor
      , descripcion , fecha , codigo , media_real ,
     desviacion_real , alfa , Rpb , 1 ) ;
6 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
7 fprintf ( Archivo_Latex , "begin{frame}{An'alisis
     General}\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex ) ;
2  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
3  }
4  void Beamer_Histograma_Notas ( FILE * Archivo_Latex ,
       long double media_real , long double
       desviacion_real , long double media_prediccion ,
       long double desviacion_prediccion )
5  {
6  long double width ;
7  width = media_real - Nota_minima ;
8  if ( ( Nota_maxima - media_real ) > width ) width =
       Nota_maxima - media_real ;
9  Prepara_Histograma_Notas ( ) ;
10 Prepara_Grafico_Normal ( media_real , desviacion_real ,
       media_prediccion , desviacion_prediccion , width )
       ;
11 fprintf ( Archivo_Latex , "begin{frame}{Histograma de
       Notas}\n" ) ;
12 fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "includegraphics[scale=0.8]{EX4010-h.
       pdf}\n" ) ;
2  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
3  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
4  fprintf ( Archivo_Latex , "begin{frame}{Distribuci'{o}n
        Normal (Predicci'on y Real)}\n" ) ;
5  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
6  fprintf ( Archivo_Latex , "centering\n" ) ;
7  fprintf ( Archivo_Latex , "includegraphics[scale=0.8]{
       EX4010-n.pdf}\n" ) ;
8  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
9  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
10 }
11 void Beamer_Dificultad_vs_Discriminacion ( FILE *
       Archivo_Latex )
12 {
13 Dificultad_vs_Discriminacion ( ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{frame}{Cruce de Discriminaci'
       on ($r_{pb}$) y Dificultad ($p$)}" ) ;
2  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
3  fprintf ( Archivo_Latex , "includegraphics[scale=0.65]{
       EX4010p.pdf}\n" ) ;
4  fprintf ( Archivo_Latex , "end{figure}\n\n" ) ;
5  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
6  fprintf ( Archivo_Latex , "begin{frame}{L'ineas de
       Contorno del Cruce entre Coeficiente de
       Discriminaci'on ($r_{pb}$) y Dificultad ($p$)}" ) ;
7  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
8  fprintf ( Archivo_Latex , "includegraphics[scale=0.65]{
       EX4010c.pdf}\n" ) ;
9  fprintf ( Archivo_Latex , "end{figure}\n\n" ) ;
10 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
11 }
12 void Beamer_Histograma_Temas ( FILE * Archivo_Latex )
13 {
14 Prepara_Histograma_Temas
```

Código Preprocesado (Sin Pretty Print)

```
1  ( ) ;
2  fprintf ( Archivo_Latex , "begin{frame}{An'{a}lisis por
       Temas (ordenado por rendimiento)}\n" ) ;
3  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
4  fprintf ( Archivo_Latex , "includegraphics[scale=0.67,
       angle=-90]{EX4010-t.pdf}\n" ) ;
5  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
6  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
7  }
8  void Beamer_Preguntas ( FILE * Archivo_Latex ,
       GtkWidget * PB , long double base , long double
       limite )
9  {
10 if ( ! gtk_toggle_button_get_active ( CK_general ) )
11 {
12 Asigna_Banderas ( ) ;
13 Lista_de_Preguntas_Beamer ( Archivo_Latex , PB , base ,
       limite ) ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  }
3  void Beamer_Gracias ( FILE * Archivo_Latex )
4  {
5  fprintf ( Archivo_Latex , "begin{frame}[plain]\n" ) ;
6  fprintf ( Archivo_Latex , "centering\n" ) ;
7  fprintf ( Archivo_Latex , "{Huge Muchas gracias}\n" ) ;
8  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
9  }
10 void Beamer_Cierre ( FILE * Archivo_Latex )
11 {
12 fprintf ( Archivo_Latex , "end{document}\n" ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Beamer_Failure ( )
2  {
3  gtk_widget_set_sensitive ( window1 , 0 ) ;
4  gtk_widget_show ( window5 ) ;
5  }
6  void Graba_Ajustes ( )
7  {
8  int i , k ;
9  char PG_command [ 3000 ] ;
10 PGresult * res ;
11 char examen [ 10 ] ;
12 k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
13 sprintf ( examen , "%05d" , k ) ;
14 res
```

Código Preprocesado (Sin Pretty Print)

```
1  = PQEXEC ( DATABASE , "BEGIN" ) ; PQclear ( res ) ;
2  sprintf ( PG_command , "DELETE from EX_examenes_ajustes
       where examen = '%s'" , examen ) ;
3  res = PQEXEC ( DATABASE , PG_command ) ; PQclear ( res
     ) ;
4  for ( i = 0 ; i < N_preguntas ; i ++ )
5  {
6  Calcula_ajuste ( i ) ;
7  if ( preguntas [ i ] . ajuste !=  0 )
8  {
9  sprintf ( PG_command , "INSERT into EX_examenes_ajustes
       values (\'%s\', %d, \'%s\', %d, %d, %d, %d, %d, %d
     , %d)" ,
10 examen , i + 1 , preguntas [ i ] . pregunta , preguntas
       [ i ] . ajuste ,
11 preguntas [ i ] . correctas_nuevas [ 0 ] , preguntas [
     i ] . correctas_nuevas [ 1 ] , preguntas [ i ] .
     correctas_nuevas [ 2 ] ,
12 preguntas [ i ] . correctas_nuevas [ 3 ] , preguntas [
```

Código Preprocesado (Sin Pretty Print)

```
= PQEXEC ( DATABASE , PG_command ) ; PQclear ( res ) ;
}
}
res = PQEXEC ( DATABASE , "END" ) ; PQclear ( res ) ;
}
void Calcula_ajuste ( int i )
{
preguntas [ i ] . ajuste = preguntas [ i ] .
    revision_especial +
( preguntas [ i ] . excluir << 8 ) +
( preguntas [ i ] . encoger << 9 ) +
( preguntas [ i ] . verbatim << 10 ) +
( preguntas [ i ] . header_encoger << 11 ) +
( preguntas [ i ] . header_verbatim << 12 ) +
(
```

Código Preprocesado (Sin Pretty Print)

```
1  preguntas [ i ] . slide [ 0 ] << 13 ) +
2  ( preguntas [ i ] . encoger_opcion [ 0 ] << 14 ) +
3  ( preguntas [ i ] . verbatim_opcion [ 0 ] << 15 ) +
4  ( preguntas [ i ] . slide [ 1 ] << 16 ) +
5  ( preguntas [ i ] . encoger_opcion [ 1 ] << 17 ) +
6  ( preguntas [ i ] . verbatim_opcion [ 1 ] << 18 ) +
7  ( preguntas [ i ] . slide [ 2 ] << 19 ) +
8  ( preguntas [ i ] . encoger_opcion [ 2 ] << 20 ) +
9  ( preguntas [ i ] . verbatim_opcion [ 2 ] << 21 ) +
10 ( preguntas [ i ] . slide [ 3 ] << 22 ) +
11 ( preguntas [ i ] . encoger_opcion [ 3 ] << 23 ) +
12 ( preguntas [ i ] . verbatim_opcion [ 3 ] << 24 ) +
13 ( preguntas [ i ] . slide [ 4 ] << 25 ) +
14 (
```

Código Preprocesado (Sin Pretty Print)

```
1  preguntas [ i ] . encoger_opcion [ 4 ] << 26 ) +
2  ( preguntas [ i ] . verbatim_opcion [ 4 ] << 27 ) ;
3  }
4  void Analisis_General ( FILE * Archivo_Latex , long
       double alfa , long double Rpb , int
       Beamer_o_reporte )
5  {
6  char mensaje [ 2000 ] ;
7  if ( alfa > 0.9 )
8  {
9  sprintf ( mensaje , "El examen muestra una textbf{
       excelente consistencia interna} ($alpha$ de
       Cronbach = textbf{%Lf}). Diversos 'itemes que miden
        la misma caracter'istica muestran un
       comportamiento bastante similar." , alfa ) ;
10 Cajita_con_bandera ( Archivo_Latex , mensaje , 0 ,
       Beamer_o_reporte ) ;
11 }
12 else
```

Código Preprocesado (Sin Pretty Print)

```
1
2 sprintf ( mensaje , "El examen muestra una textbf{buena
      consistencia interna} ($alpha$ de Cronbach =
     textbf{%Lf}). Diversos 'itemes que miden la misma
     caracter'istica muestran comportamientos similares.
    " , alfa ) ;
3 Cajita_con_bandera ( Archivo_Latex , mensaje , 1 ,
     Beamer_o_reporte ) ;
4 }
5 else
6 if ( alfa > 0.7 )
7 {
8 sprintf ( mensaje , "La textbf{consistencia interna}
     del examen es aceptable, sin ser muy buena ($alpha$
      de Cronbach = textbf{%Lf}). Se nota que el
     comportamiento de diversos 'itemes del examen es
     divergente." , alfa ) ;
9 Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
     Beamer_o_reporte ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , mensaje , 4 , Beamer_o_reporte ) ;
2  }
3  if ( Rpb > 0.3 )
4  {
5  sprintf ( mensaje , "La discriminaci'on promedio de los
       'itemes es bastante alta (textbf{%Lf}). El examen
       es muy preciso en distinguir entre estudiantes de
       buen rendimiento y bajo rendimiento." , Rpb ) ;
6  Cajita_con_bandera ( Archivo_Latex , mensaje , 0 ,
       Beamer_o_reporte ) ;
7  }
8  else
9  if ( Rpb > 0.15 )
10 {
11 sprintf ( mensaje , "El examen muestra una buena
       discriminaci'on promedio (textbf{%Lf}). Distingue
       aceptablemente entre estudiantes de alto
       rendimiento y bajo rendimiento." , Rpb ) ;
12 Cajita_con_bandera ( Archivo_Latex , mensaje , 1 ,
```

Código Preprocesado (Sin Pretty Print)

```
1
2  if ( Rpb > 0.0 )
3  {
4  sprintf ( mensaje , "La discriminaci'on promedio del
       examen es muy baja (textbf{%Lf}) aunque positiva.
       Se recomienda una revisi'on general de los
       enunciados y las opciones." , Rpb ) ;
5  Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
       Beamer_o_reporte ) ;
6  }
7  else
8  {
9  sprintf ( mensaje , "El promedio de discriminaci'on de
       las preguntas de este examen es negativo (textbf{%
       Lf}). Es indispensable hacer una revisi n
       detallada de todos los enunciados y sus opciones."
       , Rpb ) ;
10 Cajita_con_bandera ( Archivo_Latex , mensaje , 4 ,
       Beamer_o_reporte ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int i , j , N_distractores_usados ;
3  long double alfa ;
4  alfa = atof ( gtk_editable_get_chars ( GTK_EDITABLE (
       EN_alfa_real ) , 0 , -1 ) ) ;
5  for ( i = 0 ; i < N_preguntas ; i ++ )
6  {
7  for ( j = 0 ; j <  16 ; j ++ ) preguntas [ i ] . flags
       [ j ] = 0 ;
8  if ( ! preguntas [ i ] . buenos ) preguntas [ i ] .
       flags [  0 ] = 1 ;
9  if ( ! preguntas [ i ] . malos ) preguntas [ i ] .
       flags [  1 ] = 1 ;
10 if ( ( preguntas [ i ] . porcentaje < 0.31 ) &&
       preguntas [ i ] . buenos ) preguntas [ i ] . flags
       [  2 ] = 1 ;
11 if ( preguntas [ i ] . Rpb < 0.0 ) preguntas [ i ] .
       flags [  5 ] = 1 ;
12 if ( preguntas [ i ] . Rpb >= 0.3 ) preguntas [ i ] .
```

Código Preprocesado (Sin Pretty Print)

```
1  ( ( preguntas [ i ] . previo - preguntas [ i ] .
       porcentaje ) > preguntas [ i ] . desviacion )
       preguntas [ i ] . flags [  7 ] = 1 ;
2  if ( ( preguntas [ i ] . alfa_sin - alfa ) > 0.1 )
       preguntas [ i ] . flags [  8 ] = 1 ;
3  else if ( ( preguntas [ i ] . alfa_sin - alfa ) > 0.04
       ) preguntas [ i ] . flags [  9 ] = 1 ;
4  N_distractores_usados = 0 ;
5  for ( j = 0 ; j < 5 ; j ++ )
6  {
7  if ( ( preguntas [ i ] . Rpb_opcion [ j ] > 0.3 ) && (
       preguntas [ i ] . correcta != ( 'A' + j ) ) )
8  preguntas [ i ] . flags [  10 ] = 1 ;
9  else
10 if ( ( preguntas [ i ] . Rpb_opcion [ j ] > 0.0 ) && (
       preguntas [ i ] . correcta != ( 'A' + j ) ) )
11 preguntas [ i ] . flags [  11 ] = 1 ;
12 else
13 if ( ( preguntas [ i ] . Rpb_opcion [ j ] < - 0.29 ) &&
```

Código Preprocesado (Sin Pretty Print)

```
1  [ i ] . flags [  12 ] = 1 ;
2  if ( preguntas [ i ] . acumulado_opciones [ j ] > 0 )
       N_distractores_usados ++ ;
3  }
4  if ( N_distractores_usados > 3 ) preguntas [ i ] .
       flags [  13 ] = 1 ;
5  if ( ( preguntas [ i ] . Rpb >= 0.3 ) && ( preguntas [
       i ] . porcentaje <= 0.75 ) && N_distractores_usados
        <= 3 ) preguntas [ i ] . flags [  14 ] = 1 ;
6  if ( ( preguntas [ i ] . Rpb >= 0.3 ) && ( preguntas [
       i ] . porcentaje <= 0.75 ) && N_distractores_usados
        > 3 ) preguntas [ i ] . flags [  15 ] = 1 ;
7  }
8  }
9  void Color_Fila ( FILE * Archivo_Latex , int flags [
       16 ] )
10 {
11 if ( flags [  0 ] || flags [  5 ] || flags [  8 ] ||
       flags [  7 ] )
```

Código Preprocesado (Sin Pretty Print)

```
1  ( flags [  2 ] || flags [  9 ] )
2  fprintf ( Archivo_Latex , "rowcolor{yellow}\n" ) ;
3  else
4  if ( flags [  1 ] || flags [  6 ] || flags [  3 ] )
5  fprintf ( Archivo_Latex , "rowcolor{blue}\n" ) ;
6  }
7  void Lista_de_Preguntas ( FILE * Archivo_Latex ,
       GtkWidget * PB , long double base , long double
       limite )
8  {
9  int i , actual ;
10 char ejercicio_actual [  7 ] = "00000" ;
11 char PG_command [ 3000 ] ;
12 PGresult * res ;
13 char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  N_preguntas_ejercicio ;
2  actual = 0 ;
3  strcpy ( ejercicio_actual , preguntas [ actual ] .
       ejercicio ) ;
4  N_preguntas_ejercicio = 0 ;
5  for ( i = actual ; ( i < N_preguntas ) && ( strcmp (
       ejercicio_actual , preguntas [ i ] . ejercicio ) ==
        0 ) ; i ++ ) N_preguntas_ejercicio ++ ;
6  sprintf ( PG_command , "SELECT usa_header, texto_header
        from bd_texto_ejercicios, bd_ejercicios where
       codigo_ejercicio = '%s' and texto_ejercicio =
       consecutivo_texto" ,
7  ejercicio_actual ) ;
8  res = PQEXEC ( DATABASE , PG_command ) ;
9  if ( N_preguntas_ejercicio > 1 ) fprintf (
       Archivo_Latex , "\n\nrule{8.1cm}{5pt}\n\n" ) ;
10 if ( * PQgetvalue ( res , 0 , 0 ) == 't' )
11 {
12 if ( N_preguntas_ejercicio > 1 )
```

Código Preprocesado (Sin Pretty Print)

```
1  ( N_preguntas_ejercicio == 2 )
2  fprintf ( Archivo_Latex , "textbf{Las preguntas %d y %d
       requieren la siguiente informaci'{o}n:}\n\n \n\n"
     , actual + 1 , actual + 2 ) ;
3  else
4  fprintf ( Archivo_Latex , "textbf{Las preguntas %d a %d
       requieren la siguiente informaci'{o}n:}\n\n \n\n"
     , actual + 1 , actual + N_preguntas_ejercicio ) ;
5  strcpy ( hilera_antes , PQgetvalue ( res , 0 , 1 ) ) ;
6  hilera_LATEX ( hilera_antes , hilera_despues ) ;
7  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
8  fprintf ( Archivo_Latex , "rule{8.9cm}{1pt}\n" ) ;
9  fprintf ( Archivo_Latex , "begin{questions}\n" ) ;
10 }
11 else
12 {
13 fprintf ( Archivo_Latex , "begin{questions}\n" ) ;
14 Imprime_pregunta
```

Código Preprocesado (Sin Pretty Print)

```
1  ( actual , Archivo_Latex , PQgetvalue ( res , 0 , 1 ) )
       ;
2  actual ++ ;
3  N_preguntas_ejercicio = 0 ;
4  }
5  }
6  else
7  {
8  fprintf ( Archivo_Latex , "begin{questions}\n" ) ;
9  }
10 PQclear ( res ) ;
11 for ( i = 0 ; i < N_preguntas_ejercicio ; i ++ )
12 {
13 Imprime_pregunta ( actual + i , Archivo_Latex , " " ) ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  if ( N_preguntas_ejercicio > 1 ) fprintf (
       Archivo_Latex , "\n\nrule{8.1cm}{5pt}\n\n" ) ;
3  actual += N_preguntas_ejercicio ;
4  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.5 + ( ( long double ) actual /
       N_preguntas * 0.3 ) ) ;
5  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
6  while ( actual < N_preguntas )
7  {
8  strcpy ( ejercicio_actual , preguntas [ actual ] .
       ejercicio ) ;
9  N_preguntas_ejercicio = 0 ;
10 for ( i = actual ; ( i < N_preguntas ) && ( strcmp (
       ejercicio_actual , preguntas [ i ] . ejercicio ) ==
       0 ) ; i ++ ) N_preguntas_ejercicio ++ ;
11 if ( N_preguntas_ejercicio > 1 ) fprintf (
       Archivo_Latex , "\n\nrule{8.1cm}{5pt}\n\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
= PQEXEC ( DATABASE , PG_command ) ;
if ( * PQgetvalue ( res , 0 , 0 ) == 't' )
{
if ( N_preguntas_ejercicio > 1 )
{
if ( N_preguntas_ejercicio == 2 )
fprintf ( Archivo_Latex , "textbf{Las preguntas %d y %d
     requieren la siguiente informaci'{o}n:}\n\n \n\n"
    , actual + 1 , actual + 2 ) ;
else
fprintf ( Archivo_Latex , "textbf{Las preguntas %d a %d
     requieren la siguiente informaci'{o}n:}\n\n \n\n"
    , actual + 1 , actual + N_preguntas_ejercicio ) ;
strcpy ( hilera_antes , PQgetvalue ( res , 0 , 1 ) ) ;
hilera_LATEX ( hilera_antes , hilera_despues ) ;
fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
fprintf ( Archivo_Latex , "rule{8cm}{1pt}\n" ) ;
}
```

Código Preprocesado (Sin Pretty Print)

```
1
2  else
3  {
4  Imprime_pregunta ( actual , Archivo_Latex , PQgetvalue
       ( res , 0 , 1 ) ) ;
5  N_preguntas_ejercicio = 0 ;
6  actual ++ ;
7  }
8  }
9  for ( i = 0 ; i < N_preguntas_ejercicio ; i ++ )
10 {
11 Imprime_pregunta ( actual + i , Archivo_Latex , " " ) ;
12 }
13 if ( N_preguntas_ejercicio > 1 ) fprintf (
       Archivo_Latex , "\n\nrule{8.1cm}{5pt}\n\n" ) ;
14 actual
```

Código Preprocesado (Sin Pretty Print)

```
1  += N_preguntas_ejercicio ;
2  gtk_progress_bar_set_fraction ( GTK_PROGRESS_BAR (
       PB_analisis ) , 0.5 + ( ( long double ) actual /
       N_preguntas * 0.3 ) ) ;
3  while ( gtk_events_pending ( ) ) gtk_main_iteration ( )
       ;
4  }
5  fprintf ( Archivo_Latex , "end{questions}\n\n" ) ;
6  }
7  void Imprime_pregunta ( int i , FILE * Archivo_Latex ,
       char * prefijo )
8  {
9  long double cota_inferior ;
10 char hilera_antes [ 4000 ] , hilera_despues [ 4000 ] ;
11 char PG_command [ 2000 ] ;
12 PGresult * res ;
13 long double Por_A , Por_B , Por_C , Por_D , Por_E ,
       Total ;
14 gchar
```

Código Preprocesado (Sin Pretty Print)

```
1  * materia ;
2  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
3  cota_inferior =  1.15  /  6.65  * 100.0 ;
4  sprintf ( PG_command , "SELECT texto_pregunta ,
       texto_opcion_A , texto_opcion_B , texto_opcion_C ,
       texto_opcion_D , texto_opcion_E from
       bd_texto_preguntas where codigo_unico_pregunta = '%
       s'" , preguntas [ i ] . pregunta ) ;
5  res = PQEXEC ( DATABASE , PG_command ) ;
6  sprintf ( hilera_antes , "question %s\n%s" , prefijo ,
       PQgetvalue ( res , 0 , 0 ) ) ;
7  hilera_LATEX ( hilera_antes , hilera_despues ) ;
8  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
9  Por_A = ( long double ) preguntas [ i ] .
       acumulado_opciones [ 0 ] / N_estudiantes * 100.0 ;
10 Por_B = ( long double ) preguntas [ i ] .
       acumulado_opciones [ 1 ] / N_estudiantes * 100.0 ;
11 Por_C = ( long double ) preguntas [ i ] .
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{answers}\n" ) ;
2  Imprime_Opcion ( Archivo_Latex , res , Por_A , i , 0 )
       ;
3  Imprime_Opcion ( Archivo_Latex , res , Por_B , i , 1 )
       ;
4  Imprime_Opcion ( Archivo_Latex , res , Por_C , i , 2 )
       ;
5  Imprime_Opcion ( Archivo_Latex , res , Por_D , i , 3 )
       ;
6  Imprime_Opcion ( Archivo_Latex , res , Por_E , i , 4 )
       ;
7  fprintf ( Archivo_Latex , "end{answers}\n" ) ;
8  Analiza_Banderas ( Archivo_Latex , preguntas [ i ] , 0
       , 0 , 0 , NULL ) ;
9  Analiza_Ajuste ( Archivo_Latex , preguntas [ i ] , 0 )
       ;
10 fprintf ( Archivo_Latex , "framebox[7.5cm][l]{%.6s.%d -
       %.6s - %s %s %s}\n\n" ,
11 preguntas [ i ] . ejercicio , preguntas [ i ] .
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "%s\n\n" , hilera_despues ) ;
2  fprintf ( Archivo_Latex , "rule{8cm}{1pt}\n" ) ;
3  fprintf ( Archivo_Latex , "\n\n" ) ;
4  g_free ( materia ) ;
5  }
6  void Imprime_Opcion ( FILE * Archivo_Latex , PGresult *
       res , long double Porcentaje , int pregunta , int
     opcion )
7  {
8  char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
9  fprintf ( Archivo_Latex , "item " ) ;
10 strcpy ( hilera_antes , PQgetvalue ( res , 0 , 1 +
     opcion ) ) ;
11 hilera_LATEX ( hilera_antes , hilera_despues ) ;
12 fprintf ( Archivo_Latex , "%s" , hilera_despues ) ;
13 fprintf ( Archivo_Latex , "\n\n{color{green} rule{%Lfcm
     }{5pt}\n\n{footnotesize textbf{texttt{%6.2Lf %%}}}}
      (textbf{%d})\n\n" ,
14 0.05
```

Código Preprocesado (Sin Pretty Print)

```
1  +  6.65  * Porcentaje / 100.0 , Porcentaje ,
2  preguntas [ pregunta ] . acumulado_opciones [ opcion ]
       ) ;
3  fprintf ( Archivo_Latex , " $r_{pb}$ = textbf{%Lf}\n\n"
       , preguntas [ pregunta ] . Rpb_opcion [ opcion ] )
       ;
4  if ( preguntas [ pregunta ] . correcta == ( 'A' +
       opcion ) )
5  {
6  fprintf ( Archivo_Latex , "setlength{fboxrule}{2
       fboxrule}\n" ) ;
7  fprintf ( Archivo_Latex , "\n\nfcolorbox{black}{blue}{
       color{white} $starstarstar$ textbf{CORRECTA}
       $starstarstar$}" ) ;
8  fprintf ( Archivo_Latex , "setlength{fboxrule}{0.5
       fboxrule}\n" ) ;
9  }
10 fprintf ( Archivo_Latex , "\n\n" ) ;
11 }
```

Código Preprocesado (Sin Pretty Print)

```
1  N ;
2  int Ajuste ;
3  long double alfa ;
4  char mensaje [ 1000 ] ;
5  Ajuste = ( item . revision_especial !=  0 ) ;
6  N = Ajuste ;
7  alfa = atof ( gtk_editable_get_chars ( GTK_EDITABLE (
       EN_alfa_real ) , 0 , -1 ) ) ;
8  if ( item . flags [  3 ] )
9  {
10 sprintf ( mensaje , "Esta pregunta muestra un buen '{i}
       ndice de discrimaci'{o}n ($r_{pb}$ = textbf{%6.4Lf
       }). Separa muy bien a los estudiantes de buen
       rendimiento de los de bajo rendimiento." ,
11 item . Rpb ) ;
12 Cajita_con_bandera ( Archivo_Latex , mensaje , 1 ,
       beamer ) ;
13 N ++ ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
2  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
3  }
4  if ( item . flags [  12 ] )
5  {
6  sprintf ( mensaje , "Hay al menos un textbf{distractor}
       con un $r_{pb}$ muy negativo, que atrajo a los
       estudiantes de bajo rendimiento y no fue
       considerado por los estudiantes de buen rendimiento
       . textbf{Buen distractor}." ) ;
7  Cajita_con_bandera ( Archivo_Latex , mensaje , 2 ,
       beamer ) ;
8  N ++ ;
9  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
10 Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( mensaje , "Por lo menos 4 opciones diferentes fueron
       escogidas por los estudiantes." ) ;
2  Cajita_con_bandera ( Archivo_Latex , mensaje , 2 ,
       beamer ) ;
3  N ++ ;
4  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
5  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
6  }
7  if ( item . flags [ 6 ] )
8  {
9  sprintf ( mensaje , "La media de respuestas correctas a
        esta pregunta (textbf{%6.4Lf}) textbf{supera} en m
       '{a}s de una desviaci'{o}n est'{a}ndar (textbf{%6.4
       Lf}) a la media hist'{o}rica o estimada (textbf
       {%6.4Lf})." ,
10 item . porcentaje , item . desviacion , item . previo )
        ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , i , Descripcion ) ;
2  }
3  if ( item . flags [ 1 ] )
4  {
5  sprintf ( mensaje , "textbf{Todos} los estudiantes
       contestaron correctamente, por lo que la pregunta
       no discrimina de manera efectiva." ) ;
6  Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
       beamer ) ;
7  N ++ ;
8  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
9  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
10 }
11 if ( item . flags [ 2 ] )
12 {
13 sprintf ( mensaje , "S'{o}lo el textbf{%5.2Lf}%% de los
       estudiantes contestaron correctamente esta
```

Código Preprocesado (Sin Pretty Print)

```
1  . porcentaje * 100 ) ;
2  Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
       beamer ) ;
3  N ++ ;
4  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
5  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
6  }
7  if ( item . flags [ 7 ] )
8  {
9  sprintf ( mensaje , "La media de respuestas correctas a
       esta pregunta (textbf{%6.4Lf}) es textbf{menor} en
       m'{a}s de una desviaci'{o}n est'{a}ndar (textbf
       {%6.4Lf}) a la media hist'{o}rica o estimada (
       textbf{%6.4Lf})." ,
10 item . porcentaje , item . desviacion , item . previo )
       ;
11 Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , i , Descripcion ) ;
2  }
3  if ( item . flags [ 9 ] )
4  {
5  sprintf ( mensaje , "Si esta pregunta se elimina del
       examen, el $alpha$ de Cronbach subir'{i}a de textbf
       {%6.4Lf} a textbf{%6.4Lf}." ,
6  alfa , item . alfa_sin ) ;
7  Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
       beamer ) ;
8  N ++ ;
9  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
10 Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
11 }
12 if ( item . flags [ 11 ] )
13 {
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1 ( mensaje , "Hay al menos un textbf{distractor} con un
      $r_{pb}$ ligeramente positivo. Revisar enunciado y
      opciones." ) ;
2 Cajita_con_bandera ( Archivo_Latex , mensaje , 3 ,
      beamer ) ;
3 N ++ ;
4 if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
      Ajuste ) )
5 Continuar_banderas ( Archivo_Latex , i , Descripcion )
      ;
6 }
7 if ( item . flags [ 5 ] )
8 {
9 sprintf ( mensaje , "Esta pregunta muestra un '{i}ndice
       de discrimaci'{o}n ($r_{pb}$ = textbf{%6.4Lf})
      negativo. Los estudiantes de buen rendimiento en
      este examen tendieron a equivocarse, mientras que
      los de bajo rendimiento tendieron a contestarla
      bien. textbf{Revisar muy bien el enunciado y las
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , i , Descripcion ) ;
2  }
3  if ( item . flags [ 0 ] )
4  {
5  sprintf ( mensaje , "textbf{Todos} los estudiantes
       contestaron equivocadamente esta pregunta. La
       pregunta no discrimina de manera efectiva y
       posiblemente est'a mal redactada." ) ;
6  Cajita_con_bandera ( Archivo_Latex , mensaje , 4 ,
       beamer ) ;
7  N ++ ;
8  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
9  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
10 }
11 if ( item . flags [ 4 ] )
12 {
13 sprintf ( mensaje , "Esta pregunta muestra un '{i}ndice
```

Código Preprocesado (Sin Pretty Print)

```
1  . Rpb ) ;
2  Cajita_con_bandera ( Archivo_Latex , mensaje , 4 ,
       beamer ) ;
3  N ++ ;
4  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
5  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
6  }
7  if ( item . flags [ 8 ] )
8  {
9  sprintf ( mensaje , "Si esta pregunta se elimina del
       examen, el $alpha$ de Cronbach subir'{i}a textbf{
       considerablemente} (de textbf{%6.4Lf} a textbf{%6.4
       Lf})." ,
10 alfa , item . alfa_sin ) ;
11 Cajita_con_bandera ( Archivo_Latex , mensaje , 4 ,
       beamer ) ;
12 N ++ ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , i , Descripcion ) ;
2  }
3  if ( item . flags [ 10 ] )
4  {
5  sprintf ( mensaje , "Hay al menos un textbf{distractor}
       con un $r_{pb}$ muy positivo, esto significa que,
       pese a ser incorrecto, atrajo a los estudiantes de
       mejor rendimiento. textbf{Revisar cuidadosamente
       enunciado y opciones}." ) ;
6  Cajita_con_bandera ( Archivo_Latex , mensaje , 4 ,
       beamer ) ;
7  N ++ ;
8  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
9  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
10 }
11 if ( item . flags [ 14 ] )
12 {
```

Código Preprocesado (Sin Pretty Print)

```
1  . porcentaje , item . Rpb ) ;
2  Cajita_con_bandera ( Archivo_Latex , mensaje , 1 ,
       beamer ) ;
3  N ++ ;
4  if ( beamer && ( ( N % 3 ) == 0 ) && N < ( N_flags +
       Ajuste ) )
5  Continuar_banderas ( Archivo_Latex , i , Descripcion )
       ;
6  }
7  if ( item . flags [ 15 ] )
8  {
9  sprintf ( mensaje , "Esta pregunta no es f'acil (textit
       {p} = textbf{%6.4Lf}), muestra un buen 'indice de
       discriminaci'on ($r_{pb}$ = textbf{%6.4Lf}), y los
       estudiantes usaron al menos 4 de las opciones.
       textbf{Excelente pregunta}\n" ,
10 item . porcentaje , item . Rpb ) ;
11 Cajita_con_bandera ( Archivo_Latex , mensaje , 0 ,
       beamer ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , i , Descripcion ) ;
2  }
3  }
4  void Continuar_banderas ( FILE * Archivo_Latex , int i
       , char * Descripcion )
5  {
6  char hilera_antes [ 1000 ] ;
7  char hilera_despues [ 1000 ] ;
8  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
9  fprintf ( Archivo_Latex , "}\n" ) ;
10 fprintf ( Archivo_Latex , "{\n" ) ;
11 sprintf ( hilera_antes , "begin{frame}{An lisis de
       Pregunta %d - cont. hfill {small %s}}" , i ,
       Descripcion ) ;
12 hilera_LATEX ( hilera_antes , hilera_despues ) ;
13 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  void Analiza_Ajuste ( FILE * Archivo_Latex , struct
       PREGUNTA item , int modo )
3  {
4  if ( item . revision_especial == 1 )
5  Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
       Pregunta no ser'a tomada en cuenta para la evaluaci
       'on." , 5 , modo ) ;
6  if ( item . revision_especial == 2 )
7  Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
       Se considerar'an varias opciones como correctas." ,
         5 , modo ) ;
8  if ( item . revision_especial == 3 )
9  Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
       Debido a problemas en su formulaci'on, esta
       pregunta se da como textbf{correcta} a todos los
       estudiantes." , 5 , modo ) ;
10 if ( item . revision_especial == 4 )
11 Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
```

Código Preprocesado (Sin Pretty Print)

```
1  ( item . revision_especial == 6 )
2  Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
       Esta pregunta otorga un bono a los estudiantes que
       la contestaron correctamente." , 5 , modo ) ;
3  if ( item . revision_especial == 7 )
4  Cajita_con_bandera ( Archivo_Latex , "textbf{Ajuste:}
       Esta pregunta es textbf{extra} al examen, dando
       puntos adicionales sobre 100 a los que la contesten
        correctamente." , 5 , modo ) ;
5  }
6  void Cajita_con_bandera ( FILE * Archivo_Latex , char *
       mensaje , int color , int modo )
7  {
8  int i ;
9  Banderas [ color ] ++ ;
10 fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
11 fprintf ( Archivo_Latex , "centering\n" ) ;
12 fprintf ( Archivo_Latex , "setlength{fboxrule}{4
       fboxrule}\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{minipage}{1.4 cm}\n" ) ;
2  if ( color == 5 )
3  fprintf ( Archivo_Latex , "includegraphics[scale
       =0.27]{.imagenes/%s}\n" , banderas [ color ] ) ;
4  else
5  fprintf ( Archivo_Latex , "includegraphics[scale=0.07,
       angle=45]{.imagenes/%s}\n" , banderas [ color ] ) ;
6  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
7  if ( modo == 1 )
8  fprintf ( Archivo_Latex , "begin{minipage}{8.6 cm}\n" )
       ;
9  else
10 fprintf ( Archivo_Latex , "begin{minipage}{6.1 cm}\n" )
       ;
11 fprintf ( Archivo_Latex , "small{%s}\n" , mensaje ) ;
12 fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
13 fprintf ( Archivo_Latex , "}\n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "setlength{fboxrule}{0.25fboxrule}\n"
       ) ;
2  fprintf ( Archivo_Latex , "end{figure}\n" ) ;
3  }
4  void Resumen_de_Banderas ( FILE * Archivo_Latex , int
       modo )
5  {
6  int color ;
7  if ( ! modo )
8  fprintf ( Archivo_Latex , "center{textbf{Resumen de
       Observaciones}}\n\n" ) ;
9  for ( color = 0 ; color < 5 ; color ++ )
10 {
11 fprintf ( Archivo_Latex , "begin{minipage}{3.50 cm}\n"
       ) ;
12 fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
13 fprintf ( Archivo_Latex , "centering\n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "setlength{fboxrule}{4fboxrule}\n" )
       ;
2  fprintf ( Archivo_Latex , "fcolorbox{%s}{white}{\n" ,
       colores [ color ] ) ;
3  fprintf ( Archivo_Latex , "begin{minipage}{1.5 cm}\n" )
        ;
4  fprintf ( Archivo_Latex , "includegraphics[scale=0.07,
       angle=45]{.imagenes/%s}\n" , banderas [ color ] ) ;
5  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
6  fprintf ( Archivo_Latex , "begin{minipage}{0.5 cm}\n" )
        ;
7  fprintf ( Archivo_Latex , "large{textbf{%d}}\n" ,
       Banderas [ color ] ) ;
8  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
9  fprintf ( Archivo_Latex , "}\n" ) ;
10 fprintf ( Archivo_Latex , "setlength{fboxrule}{0.25
       fboxrule}\n" ) ;
11 fprintf ( Archivo_Latex , "end{figure}\n" ) ;
12 fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{minipage}{3.50 cm}\n" ) ;
2  fprintf ( Archivo_Latex , "begin{figure}[H]\n" ) ;
3  fprintf ( Archivo_Latex , "centering\n" ) ;
4  fprintf ( Archivo_Latex , "setlength{fboxrule}{4
       fboxrule}\n" ) ;
5  fprintf ( Archivo_Latex , "fcolorbox{%s}{white}{\n" ,
       colores [ 5 ] ) ;
6  fprintf ( Archivo_Latex , "begin{minipage}{1.5 cm}\n" )
       ;
7  fprintf ( Archivo_Latex , "includegraphics[scale
       =0.27]{.imagenes/%s}\n" , banderas [ 5 ] ) ;
8  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
9  fprintf ( Archivo_Latex , "begin{minipage}{0.5 cm}\n" )
       ;
10 fprintf ( Archivo_Latex , "large{textbf{%d}}\n" ,
       Banderas [ 5 ] ) ;
11 fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
12 fprintf ( Archivo_Latex , "}\n" ) ;
13 fprintf ( Archivo_Latex , "setlength{fboxrule}{0.25
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "end{figure}\n" ) ;
2  fprintf ( Archivo_Latex , "end{minipage}\n" ) ;
3  }
4  void Lista_de_Notas ( FILE * Archivo_Latex )
5  {
6  int i , k , N_estudiantes ;
7  char examen [ 10 ] ;
8  char PG_command [ 2000 ] ;
9  PGresult * res ;
10 long double Porcentaje , Porcentaje_ajustado ;
11 int N_correctas , N_ajustado , M_ajustado ;
12 char hilera_antes [ 2000 ] , hilera_despues [ 2000 ] ;
13 k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( examen , "%05d" , k ) ;
2  sprintf ( PG_command , "SELECT nombre, version,
       respuestas, correctas, porcentaje from
       EX_examenes_respuestas where examen = '%s' order by
        nombre" , examen ) ;
3  res = PQEXEC ( DATABASE , PG_command ) ;
4  N_estudiantes = PQntuples ( res ) ;
5  fprintf ( Archivo_Latex , "\n\n" ) ;
6  fprintf ( Archivo_Latex , "begin{center}\n" ) ;
7  fprintf ( Archivo_Latex , "begin{longtable}{|l|c|c|c|c|
       c|}\n" ) ;
8  fprintf ( Archivo_Latex , "hline\n" ) ;
9  fprintf ( Archivo_Latex , "textbf{Nombre} & textbf{
       Versi'{o}n} & textbf{Correctas} & textbf{Porcentaje
       } & textbf{Ajuste} & textbf{Nota Ajustada} hline
       hline\n" ) ;
10 fprintf ( Archivo_Latex , "endfirsthead\n" ) ;
11 fprintf ( Archivo_Latex , "hline\n" ) ;
12 fprintf ( Archivo_Latex , "textbf{Nombre} & textbf{
```

Código Preprocesado (Sin Pretty Print)

```
1  ( i = 0 ; i < N_estudiantes ; i ++ )
2  {
3  Calcula_Notas ( PQgetvalue ( res , i , 1 ) ,
4  PQgetvalue ( res , i , 2 ) ,
5  & N_correctas , & N_ajustado , & M_ajustado ) ;
6  Porcentaje = ( long double ) N_correctas / N_preguntas
       * 100.0 ;
7  if ( M_ajustado )
8  Porcentaje_ajustado = ( long double ) N_ajustado /
       M_ajustado * 100.0 ;
9  else
10 Porcentaje_ajustado = 0.0 ;
11 sprintf ( hilera_antes , "%s & %s & %d/%d & %7.2Lf & %d
       /%d & textbf{%7.2Lf}  hline" ,
12 PQgetvalue ( res , i , 0 ) , PQgetvalue ( res , i , 1 )
       ,
13 N_correctas , N_preguntas , Porcentaje ,
14 N_ajustado
```

Código Preprocesado (Sin Pretty Print)

```
1  , M_ajustado , Porcentaje_ajustado ) ;
2  hilera_LATEX ( hilera_antes , hilera_despues ) ;
3  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
4  }
5  fprintf ( Archivo_Latex , "end{longtable}\n" ) ;
6  fprintf ( Archivo_Latex , "end{center}\n" ) ;
7  }
8  void Calcula_Notas ( char * version , char * respuestas
       , int * n_buenas , int * n_ajustado , int *
     m_ajustado )
9  {
10 int i , k_version ;
11 int N , K , M ;
12 for ( k_version = 0 ; ( k_version < N_versiones ) &&
     strcmp ( versiones [ k_version ] . codigo , version
       ) != 0 ; k_version ++ ) ;
13 N = 0 ;
14 for
```

Código Preprocesado (Sin Pretty Print)

```
1  ( i = 0 ; i < N_preguntas ; i ++ )
2  if ( versiones [ k_version ] . preguntas [ i ] .
       respuesta == respuestas [ i ] ) N ++ ;
3  if ( N_ajustes == 0 )
4  {
5  K = N ;
6  M = N_preguntas ;
7  }
8  else
9  Calcula_nota_ajustada ( k_version , respuestas , & K ,
       & M ) ;
10 * n_buenas = N ;
11 * n_ajustado = K ;
12 * m_ajustado = M ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Calcula_nota_ajustada ( int k_version , char *
       respuestas , int * n , int * m )
2  {
3  int i , k ;
4  int N_correctas , N_evaluadas ;
5  N_correctas = N_evaluadas = 0 ;
6  for ( i = 0 ; i < N_preguntas ; i ++ )
7  {
8  for ( k = 0 ; ( k < N_preguntas ) && strcmp ( versiones
       [ k_version ] . preguntas [ i ] . codigo ,
       preguntas [ k ] . pregunta ) != 0 ; k ++ ) ;
9  switch ( preguntas [ k ] . revision_especial )
10 {
11 case  0 :
12 if ( versiones [ k_version ] . preguntas [ i ] .
       respuesta == respuestas [ i ] ) N_correctas ++ ;
13 N_evaluadas ++ ;
14 break
```

Código Preprocesado (Sin Pretty Print)

```
1  ;
2  case  1 :
3  break ;
4  case  3 :
5  N_correctas ++ ;
6  N_evaluadas ++ ;
7  break ;
8  case  2 :
9  break ;
10 case  4 :
11 N_evaluadas ++ ;
12 break ;
13 case  5 :
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( versiones [ k_version ] . preguntas [ i ] . respuesta
       == respuestas [ i ] )
2  {
3  N_correctas ++ ;
4  N_evaluadas ++ ;
5  }
6  break ;
7  case 6 :
8  if ( versiones [ k_version ] . preguntas [ i ] .
       respuesta == respuestas [ i ] ) N_correctas += 2 ;
9  N_evaluadas ++ ;
10 break ;
11 case 7 :
12 if ( versiones [ k_version ] . preguntas [ i ] .
       respuesta == respuestas [ i ] ) N_correctas ++ ;
13 break ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  }
3  * n = N_correctas ;
4  * m = N_evaluadas ;
5  }
6  void Cambia_Pregunta ( )
7  {
8  int i , k ;
9  GdkColor color ;
10 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
11 if ( k && preguntas )
12 {
13 Color_ajustes ( k - 1 ) ;
14 gtk_combo_box_set_active
```

Código Preprocesado (Sin Pretty Print)

```
1  ( CB_ajuste , preguntas [ k - 1 ] . revision_especial )
       ;
2  if ( preguntas [ k - 1 ] . revision_especial != 0 )
3  {
4  if ( preguntas [ k - 1 ] . actualizar )
5  gtk_toggle_button_set_active ( CK_no_actualiza , FALSE
       ) ;
6  else
7  gtk_toggle_button_set_active ( CK_no_actualiza , TRUE )
       ;
8  if ( preguntas [ k - 1 ] . revision_especial == 2 )
9  {
10 if ( preguntas [ k - 1 ] . correctas_nuevas [ 0 ] )
       gtk_toggle_button_set_active ( TG_A , TRUE ) ;
11 if ( preguntas [ k - 1 ] . correctas_nuevas [ 1 ] )
       gtk_toggle_button_set_active ( TG_B , TRUE ) ;
12 if ( preguntas [ k - 1 ] . correctas_nuevas [ 2 ] )
       gtk_toggle_button_set_active ( TG_C , TRUE ) ;
13 if ( preguntas [ k - 1 ] . correctas_nuevas [ 3 ] )
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ k - 1 ] . correctas_nuevas [ 4 ] )
       gtk_toggle_button_set_active ( TG_E , TRUE ) ;
2  }
3  }
4  gtk_text_buffer_set_text ( buffer_TV_pregunta ,
       preguntas [ k - 1 ] . texto_pregunta , - 1 ) ;
5  if ( preguntas [ k - 1 ] . grupo_inicio == preguntas [
       k - 1 ] . grupo_final )
6  {
7  gtk_toggle_button_set_active ( CK_header_encoger ,
       FALSE ) ;
8  gtk_toggle_button_set_active ( CK_header_verbatim ,
       FALSE ) ;
9  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_header_encoger ) , 0 ) ;
10 gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_header_verbatim ) , 0 ) ;
11 }
12 else
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_WIDGET ( CK_header_encoger ) , 1 ) ;
2  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_header_verbatim ) , 1 ) ;
3  if ( preguntas [ k - 1 ] . header_encoger )
4  gtk_toggle_button_set_active ( CK_header_encoger , TRUE
       ) ;
5  else
6  gtk_toggle_button_set_active ( CK_header_encoger ,
       FALSE ) ;
7  if ( preguntas [ k - 1 ] . header_verbatim )
8  gtk_toggle_button_set_active ( CK_header_verbatim ,
       TRUE ) ;
9  else
10 gtk_toggle_button_set_active ( CK_header_verbatim ,
       FALSE ) ;
11 }
12 if ( preguntas [ k - 1 ] . excluir )
13 gtk_toggle_button_set_active ( CK_excluir , TRUE ) ;
14 else
```

Código Preprocesado (Sin Pretty Print)

```
1
2  gtk_toggle_button_set_active ( CK_excluir , FALSE ) ;
3  if ( preguntas [ k - 1 ] . encoger )
4  gtk_toggle_button_set_active ( CK_encoger , TRUE ) ;
5  else
6  gtk_toggle_button_set_active ( CK_encoger , FALSE ) ;
7  if ( preguntas [ k - 1 ] . verbatim )
8  gtk_toggle_button_set_active ( CK_verbatim , TRUE ) ;
9  else
10 gtk_toggle_button_set_active ( CK_verbatim , FALSE ) ;
11 for ( i = 0 ; i < 5 ; i ++ )
12 {
13 if ( preguntas [ k - 1 ] . slide [ i ] )
14 gtk_toggle_button_set_active
```

Código Preprocesado (Sin Pretty Print)

```
1  ( CK_slide [ i ] , TRUE ) ;
2  else
3  gtk_toggle_button_set_active ( CK_slide [ i ] , FALSE )
       ;
4  }
5  for ( i = 0 ; i < 5 ; i ++ )
6  {
7  if ( preguntas [ k - 1 ] . encoger_opcion [ i ] )
8  gtk_toggle_button_set_active ( CK_encoger_opcion [ i ]
       , TRUE ) ;
9  else
10 gtk_toggle_button_set_active ( CK_encoger_opcion [ i ]
       , FALSE ) ;
11 }
12 for ( i = 0 ; i < 5 ; i ++ )
13 {
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ k - 1 ] . verbatim_opcion [ i ] )
2  gtk_toggle_button_set_active ( CK_verbatim_opcion [ i ]
       , TRUE ) ;
3  else
4  gtk_toggle_button_set_active ( CK_verbatim_opcion [ i ]
       , FALSE ) ;
5  }
6  }
7  }
8  void Cambio_Ajuste ( GtkWidget * widget , gpointer
       user_data )
9  {
10 int j , k , previo ;
11 GdkColor color ;
12 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
13 j = gtk_combo_box_get_active ( CB_ajuste ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( k && preguntas )
2  {
3  previo = preguntas [ k - 1 ] . revision_especial ;
4  preguntas [ k - 1 ] . revision_especial = j ;
5  if ( j != 2 )
6  {
7  gtk_toggle_button_set_active ( TG_A , FALSE ) ;
8  gtk_toggle_button_set_active ( TG_B , FALSE ) ;
9  gtk_toggle_button_set_active ( TG_C , FALSE ) ;
10 gtk_toggle_button_set_active ( TG_D , FALSE ) ;
11 gtk_toggle_button_set_active ( TG_E , FALSE ) ;
12 gtk_widget_set_sensitive ( GTK_WIDGET ( TG_A ) , 0 ) ;
13 gtk_widget_set_sensitive ( GTK_WIDGET ( TG_B ) , 0 ) ;
14 gtk_widget_set_sensitive
```

Código Preprocesado (Sin Pretty Print)

```
1  ( GTK_WIDGET ( TG_C ) , 0 ) ;
2  gtk_widget_set_sensitive ( GTK_WIDGET ( TG_D ) , 0 ) ;
3  gtk_widget_set_sensitive ( GTK_WIDGET ( TG_E ) , 0 ) ;
4  }
5  else
6  {
7  gtk_widget_set_sensitive ( GTK_WIDGET ( TG_A ) , 1 ) ;
8  gtk_widget_set_sensitive ( GTK_WIDGET ( TG_B ) , 1 ) ;
9  gtk_widget_set_sensitive ( GTK_WIDGET ( TG_C ) , 1 ) ;
10 gtk_widget_set_sensitive ( GTK_WIDGET ( TG_D ) , 1 ) ;
11 gtk_widget_set_sensitive ( GTK_WIDGET ( TG_E ) , 1 ) ;
12 }
13 if ( j ==  0 )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  gtk_toggle_button_set_active ( CK_no_actualiza , FALSE
       ) ;
3  gtk_widget_set_sensitive ( GTK_WIDGET ( CK_no_actualiza
       ) , 0 ) ;
4  preguntas [ k - 1 ] . actualizar = 1 ;
5  }
6  else
7  {
8  gtk_widget_set_sensitive ( GTK_WIDGET ( CK_no_actualiza
       ) , 1 ) ;
9  if ( previo == 0 )
10 gtk_toggle_button_set_active ( CK_no_actualiza , TRUE )
       ;
11 else
12 {
13 if ( preguntas [ k - 1 ] . actualizar )
14 gtk_toggle_button_set_active
```

Código Preprocesado (Sin Pretty Print)

```
1  ( CK_no_actualiza , FALSE ) ;
2  else
3  gtk_toggle_button_set_active ( CK_no_actualiza , TRUE )
       ;
4  }
5  }
6  Color_ajustes ( k - 1 ) ;
7  }
8  N_ajustes = 0 ;
9  for ( j = 0 ; j < N_preguntas ; j ++ ) if ( preguntas [
       j ] . revision_especial != 0 ) N_ajustes ++ ;
10 }
11 void Cambio_no_actualizar ( GtkWidget * widget ,
       gpointer user_data )
12 {
13 int k ;
14 k
```

Código Preprocesado (Sin Pretty Print)

```
1  = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
2  if ( k )
3  {
4  if ( gtk_toggle_button_get_active ( CK_no_actualiza ) )
5  preguntas [ k - 1 ] . actualizar = 0 ;
6  else
7  preguntas [ k - 1 ] . actualizar = 1 ;
8  }
9  }
10 void Cambio_excluir ( GtkWidget * widget , gpointer
       user_data )
11 {
12 int k ;
13 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( k )
2  {
3  if ( gtk_toggle_button_get_active ( CK_excluir ) )
4  preguntas [ k - 1 ] . excluir = 1 ;
5  else
6  preguntas [ k - 1 ] . excluir = 0 ;
7  Color_ajustes ( k - 1 ) ;
8  }
9  }
10 void Cambio_encoger ( GtkWidget * widget , gpointer
       user_data )
11 {
12 int k ;
13 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( k )
2  {
3  if ( gtk_toggle_button_get_active ( CK_encoger ) )
4  preguntas [ k - 1 ] . encoger = 1 ;
5  else
6  preguntas [ k - 1 ] . encoger = 0 ;
7  Color_ajustes ( k - 1 ) ;
8  }
9  }
10 void Cambio_verbatim ( GtkWidget * widget , gpointer
       user_data )
11 {
12 int k ;
13 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( k )
2  {
3  if ( gtk_toggle_button_get_active ( CK_verbatim ) )
4  preguntas [ k - 1 ] . verbatim = 1 ;
5  else
6  preguntas [ k - 1 ] . verbatim = 0 ;
7  Color_ajustes ( k - 1 ) ;
8  }
9  }
10 void Cambio_header_encoger ( GtkWidget * widget ,
       gpointer user_data )
11 {
12 int k , j ;
13 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( k )
2  {
3  if ( gtk_toggle_button_get_active ( CK_header_encoger )
       )
4  for ( j = preguntas [ k - 1 ] . grupo_inicio ; j <=
       preguntas [ k - 1 ] . grupo_final ; j ++ )
5  preguntas [ j ] . header_encoger = 1 ;
6  else
7  for ( j = preguntas [ k - 1 ] . grupo_inicio ; j <=
       preguntas [ k - 1 ] . grupo_final ; j ++ )
8  preguntas [ j ] . header_encoger = 0 ;
9  Color_ajustes ( k - 1 ) ;
10 }
11 }
12 void Cambio_header_verbatim ( GtkWidget * widget ,
       gpointer user_data )
13 {
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  k , j ;
2  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
3  if ( k )
4  {
5  if ( gtk_toggle_button_get_active ( CK_header_verbatim
       ) )
6  for ( j = preguntas [ k - 1 ] . grupo_inicio ; j <=
       preguntas [ k - 1 ] . grupo_final ; j ++ )
7  preguntas [ j ] . header_verbatim = 1 ;
8  else
9  for ( j = preguntas [ k - 1 ] . grupo_inicio ; j <=
       preguntas [ k - 1 ] . grupo_final ; j ++ )
10 preguntas [ j ] . header_verbatim = 0 ;
11 Color_ajustes ( k - 1 ) ;
12 }
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Cambio_slide ( int i )
2  {
3  int k ;
4  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
5  if ( k )
6  {
7  if ( gtk_toggle_button_get_active ( CK_slide [ i ] ) )
8  {
9  preguntas [ k - 1 ] . slide [ i ] = 1 ;
10 gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_encoger_opcion [ i ] ) , 1 ) ;
11 gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_verbatim_opcion [ i ] ) , 1 ) ;
12 }
13 else
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  preguntas [ k - 1 ] . slide [ i ] = 0 ;
3  preguntas [ k - 1 ] . encoger_opcion [ i ] = 0 ;
4  preguntas [ k - 1 ] . verbatim_opcion [ i ] = 0 ;
5  gtk_toggle_button_set_active ( CK_encoger_opcion [ i ]
       , FALSE ) ;
6  gtk_toggle_button_set_active ( CK_verbatim_opcion [ i ]
        , FALSE ) ;
7  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_encoger_opcion [ i ] ) , 0 ) ;
8  gtk_widget_set_sensitive ( GTK_WIDGET (
       CK_verbatim_opcion [ i ] ) , 0 ) ;
9  }
10 Color_ajustes ( k - 1 ) ;
11 }
12 }
13 void Cambio_encoger_opcion ( int i )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int k ;
3  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
4  if ( k )
5  {
6  if ( gtk_toggle_button_get_active ( CK_encoger_opcion [
       i ] ) )
7  preguntas [ k - 1 ] . encoger_opcion [ i ] = 1 ;
8  else
9  preguntas [ k - 1 ] . encoger_opcion [ i ] = 0 ;
10 Color_ajustes ( k - 1 ) ;
11 }
12 }
13 void Cambio_verbatim_opcion ( int i )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int k ;
3  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
4  if ( k )
5  {
6  if ( gtk_toggle_button_get_active ( CK_verbatim_opcion
       [ i ] ) )
7  preguntas [ k - 1 ] . verbatim_opcion [ i ] = 1 ;
8  else
9  preguntas [ k - 1 ] . verbatim_opcion [ i ] = 0 ;
10 Color_ajustes ( k - 1 ) ;
11 }
12 }
13 void Cambio_A ( GtkWidget * widget , gpointer user_data
       )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  int k ;
3  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
4  preguntas [ k - 1 ] . correctas_nuevas [ 0 ] =
       gtk_toggle_button_get_active ( TG_A ) ? 1 : 0 ;
5  Color_ajustes ( k - 1 ) ;
6  }
7  void Cambio_B ( GtkWidget * widget , gpointer user_data
       )
8  {
9  int k ;
10 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
11 preguntas [ k - 1 ] . correctas_nuevas [ 1 ] =
       gtk_toggle_button_get_active ( TG_B ) ? 1 : 0 ;
12 Color_ajustes ( k - 1 ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Cambio_C ( GtkWidget * widget , gpointer user_data )
2  {
3  int k ;
4  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
5  preguntas [ k - 1 ] . correctas_nuevas [ 2 ] =
       gtk_toggle_button_get_active ( TG_C ) ? 1 : 0 ;
6  Color_ajustes ( k - 1 ) ;
7  }
8  void Cambio_D ( GtkWidget * widget , gpointer user_data
       )
9  {
10 int k ;
11 k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
12 preguntas [ k - 1 ] . correctas_nuevas [ 3 ] =
       gtk_toggle_button_get_active ( TG_D ) ? 1 : 0 ;
13 Color_ajustes ( k - 1 ) ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  void Cambio_E ( GtkWidget * widget , gpointer user_data
       )
3  {
4  int k ;
5  k = ( int ) gtk_range_get_value ( GTK_RANGE (
       SC_preguntas ) ) ;
6  preguntas [ k - 1 ] . correctas_nuevas [ 4 ] =
       gtk_toggle_button_get_active ( TG_E ) ? 1 : 0 ;
7  Color_ajustes ( k - 1 ) ;
8  }
9  void Color_ajustes ( int k )
10 {
11 GdkColor color ;
12 Calcula_ajuste ( k ) ;
13 if ( preguntas [ k ] . ajuste - preguntas [ k ] .
       revision_especial > 0 )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  gdk_color_parse ( RARE_AREA , & color ) ;
3  gtk_widget_modify_bg ( EB_formato , GTK_STATE_NORMAL ,
       & color ) ;
4  }
5  else
6  {
7  gdk_color_parse ( SECONDARY_AREA , & color ) ;
8  gtk_widget_modify_bg ( EB_formato , GTK_STATE_NORMAL ,
       & color ) ;
9  }
10 if ( preguntas [ k ] . revision_especial !=  0 )
11 {
12 gdk_color_parse ( RARE_AREA , & color ) ;
13 gtk_widget_modify_bg ( EB_ajustes , GTK_STATE_NORMAL ,
       & color ) ;
14 }
```

Código Preprocesado (Sin Pretty Print)

```
1
2  else
3  {
4  gdk_color_parse ( SECONDARY_AREA , & color ) ;
5  gtk_widget_modify_bg ( EB_ajustes , GTK_STATE_NORMAL ,
       & color ) ;
6  }
7  }
8  void Fin_de_Programa ( GtkWidget * widget , gpointer
       user_data )
9  {
10 PQfinish ( DATABASE ) ;
11 gtk_main_quit ( ) ;
12 exit ( 0 ) ;
13 }
14 void
```

Código Preprocesado (Sin Pretty Print)

```
1  Fin_Ventana ( GtkWidget * widget , gpointer user_data )
2  {
3  PQfinish ( DATABASE ) ;
4  gtk_main_quit ( ) ;
5  exit ( 0 ) ;
6  }
7  void Prepara_Grafico_Normal ( long double media , long
       double desviacion , long double media_pred , long
       double desviacion_pred , long double width )
8  {
9  int i ;
10 FILE * Archivo_gnuplot ;
11 char Hilera_Antes [ 2000 ] , Hilera_Despues [ 2000 ] ;
12 char comando [ 2000 ] ;
13 Archivo_gnuplot = fopen ( "EX4010-n.gp" , "w" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set term postscript eps enhanced
       color \"Times\" 12\n" ) ;
2  fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
       ) ;
3  fprintf ( Archivo_gnuplot , "set size 0.9, 0.9\n" ) ;
4  fprintf ( Archivo_gnuplot , "set grid xtics\n" ) ;
5  fprintf ( Archivo_gnuplot , "set output \"EX4010-n.eps
       \"\n" ) ;
6  hilera_GNUPLOT ( "set ylabel \"Proporci n\"\n" ,
       Hilera_Despues ) ;
7  fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
8  hilera_GNUPLOT ( "set xlabel \"Nota\"\n" ,
       Hilera_Despues ) ;
9  fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
10 fprintf ( Archivo_gnuplot , "set xrange [%Lf:%Lf]\n" ,
       ( media - width ) - 3.0 , ( media + width ) + 3.0 )
       ;
11 fprintf ( Archivo_gnuplot , "load \".scripts/stat.inc
       \"\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set arrow from %Lf,0.0 to %Lf,
       graph(1, 1) linetype 2 lw 4 lc 1 nohead front\n" ,
       media , media ) ;
2  fprintf ( Archivo_gnuplot , "set arrow from %Lf,0.0 to
       %Lf,graph(1, 1) linetype 2 lw 4 lc 1 nohead front\n
       " , media_pred , media_pred ) ;
3  sprintf ( Hilera_Antes , "plot normal (x, %Lf, %Lf)
       with lines linetype 1 lw 7 lc 3 title \"Predicci n
       \", normal (x, %Lf, %Lf) with lines linetype 1 lw 7
        lc 2 title \"Real\"" ,
4  media_pred , desviacion_pred , media , desviacion ) ;
5  hilera_GNUPLOT ( Hilera_Antes , Hilera_Despues ) ;
6  fprintf ( Archivo_gnuplot , "%s\n" , Hilera_Despues ) ;
7  fclose ( Archivo_gnuplot ) ;
8  sprintf ( comando , "%s EX4010-n.gp" , parametros .
       gnuplot ) ;
9  system ( comando ) ;
10 sprintf ( comando , "mv EX4010-n.gp %s" , parametros .
       ruta_gnuplot ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( "rm EX4010-n.eps" ) ;
2  }
3  void Prepara_Histograma_Notas ( )
4  {
5  int i , maximo ;
6  FILE * Archivo_gnuplot , * Archivo_Datos ;
7  char Hilera_Antes [ 2000 ] , Hilera_Despues [ 2000 ] ;
8  char comando [ 2000 ] ;
9  maximo = 0 ;
10 Archivo_Datos = fopen ( "EX4010y.dat" , "w" ) ;
11 for ( i = 0 ; i <  10 ; i ++ )
12 {
13 if ( Frecuencias [ i ] ) fprintf ( Archivo_Datos , "%Lf
       %Lf\n" , ( long double ) i * ( 100.0 / ( long
      double )  10 ) , ( long double ) Frecuencias [ i ]
      ) ;
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Frecuencias [ i ] > maximo ) maximo = Frecuencias [ i
       ] ;
2  }
3  fclose ( Archivo_Datos ) ;
4  Archivo_gnuplot = fopen ( "EX4010-h.gp" , "w" ) ;
5  fprintf ( Archivo_gnuplot , "set term postscript eps
       enhanced color \"Times\" 12\n" ) ;
6  fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
       ) ;
7  fprintf ( Archivo_gnuplot , "set size 0.9, 0.9\n" ) ;
8  fprintf ( Archivo_gnuplot , "set grid xtics\n" ) ;
9  fprintf ( Archivo_gnuplot , "set output \"EX4010-h.eps
       \"\n" ) ;
10 hilera_GNUPLOT ( "set ylabel \"Cantidad\"\n" ,
       Hilera_Despues ) ;
11 fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
12 hilera_GNUPLOT ( "set xlabel \"Nota\"\n" ,
       Hilera_Despues ) ;
13 fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set xrange [0.0:100.0]\n" ) ;
2  fprintf ( Archivo_gnuplot , "set xtics 10\n" ) ;
3  fprintf ( Archivo_gnuplot , "set yrange [0.0:%d]\n" ,
       maximo + 2 ) ;
4  fprintf ( Archivo_gnuplot , "set style fill solid 1.0
       border -1\n" ) ;
5  fprintf ( Archivo_gnuplot , "set boxwidth 8\n" ) ;
6  fprintf ( Archivo_gnuplot , "plot \"EX4010y.dat\" with
       boxes fill lw 3 lc 2 notitle" ) ;
7  fclose ( Archivo_gnuplot ) ;
8  sprintf ( comando , "%s EX4010-h.gp" , parametros .
       gnuplot ) ;
9  system ( comando ) ;
10 sprintf ( comando , "mv EX4010-h.gp %s" , parametros .
       ruta_gnuplot ) ;
11 system ( comando ) ;
12 sprintf ( comando , "%s EX4010-h.eps" , parametros .
       epstopdf ) ;
13 system ( comando ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( "rm EX4010-h.eps" ) ;
2  system ( "rm EX4010y.dat" ) ;
3  }
4  void Prepara_Histograma_Temas ( )
5  {
6  int i , maximo ;
7  FILE * Archivo_gnuplot , * Archivo_Datos ;
8  char Hilera_Antes [ 2000 ] , Hilera_Despues [ 2000 ] ;
9  char PG_command [ 1000 ] ;
10 gchar * materia ;
11 PGresult * res ;
12 char comando [ 2000 ] ;
13 materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
14 Archivo_Datos
```

Código Preprocesado (Sin Pretty Print)

```
1  = fopen ( "EX4010z.dat" , "w" ) ;
2  for ( i = 0 ; i < N_temas ; i ++ )
3  {
4  fprintf ( Archivo_Datos , "%d %Lf %Lf\n" , i , ( long
       double ) ( ( resumen_tema [ i ] . buenos +
       resumen_tema [ i ] . malos ) / N_estudiantes ) /
       N_preguntas * 100.0 ,
5  ( long double ) resumen_tema [ i ] . buenos / (
       resumen_tema [ i ] . buenos + resumen_tema [ i ] .
       malos ) * 100.0 ) ;
6  }
7  fclose ( Archivo_Datos ) ;
8  Archivo_gnuplot = fopen ( "EX4010-t.gp" , "w" ) ;
9  fprintf ( Archivo_gnuplot , "set term postscript eps
       enhanced color \"Times\" 12\n" ) ;
10 fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
       ) ;
11 fprintf ( Archivo_gnuplot , "set grid y2tics ytics\n" )
       ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set style data histogram\n" ) ;
2  fprintf ( Archivo_gnuplot , "set style histogram
       cluster gap 1\n" ) ;
3  fprintf ( Archivo_gnuplot , "set style fill solid
       border -1\n" ) ;
4  fprintf ( Archivo_gnuplot , "set boxwidth 0.8\n" ) ;
5  fprintf ( Archivo_gnuplot , "set xtics (" ) ;
6  for ( i = 0 ; i < N_temas ; i ++ )
7  {
8  sprintf ( PG_command , "SELECT descripcion_materia from
        BD_materias where codigo_materia = '%s' and
       codigo_tema = '%s' and codigo_subtema = '
       '" , materia , resumen_tema [ i ] . tema ) ;
9  res = PQEXEC ( DATABASE , PG_command ) ;
10 sprintf ( Hilera_Antes , "\"%.44s (%3d)\" %d" ,
       PQgetvalue ( res , 0 , 0 ) , ( resumen_tema [ i ] .
        buenos + resumen_tema [ i ] . malos ) /
       N_estudiantes , i ) ;
11 hilera_GNUPLOT ( Hilera_Antes , Hilera_Despues ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , ")\n" ) ;
2  else
3  fprintf ( Archivo_gnuplot , ", " ) ;
4  PQclear ( res ) ;
5  }
6  fprintf ( Archivo_gnuplot , "set xtics rotate by -270
       scale 0\n" ) ;
7  fprintf ( Archivo_gnuplot , "set ytics rotate by 90\n"
       ) ;
8  fprintf ( Archivo_gnuplot , "set ytics 5,10\n" ) ;
9  fprintf ( Archivo_gnuplot , "set y2tics rotate by 90\n"
        ) ;
10 fprintf ( Archivo_gnuplot , "set y2tics 0,10\n" ) ;
11 fprintf ( Archivo_gnuplot , "set yrange [0:100.0]\n" )
       ;
12 fprintf ( Archivo_gnuplot , "set xlabel \" \"\n" ) ;
13 fprintf ( Archivo_gnuplot , "set size 0.63, 2.0\n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set label 2 \"%% preguntas\" at
       graph 0.13, 0.85 left rotate by 90\n" ) ;
2  fprintf ( Archivo_gnuplot , "set label 3 \"%% correctas
       \" at graph 0.21, 0.85 left rotate by 90\n" ) ;
3  fprintf ( Archivo_gnuplot , "plot \"EX4010z.dat\" using
        2 title \" \" lc 3 lt 1, \"\" using 3 title \" \"
       lc 2 lt 1\n" ) ;
4  fclose ( Archivo_gnuplot ) ;
5  sprintf ( comando , "%s EX4010-t.gp" , parametros .
       gnuplot ) ;
6  system ( comando ) ;
7  sprintf ( comando , "mv EX4010-t.gp %s" , parametros .
       ruta_gnuplot ) ;
8  system ( comando ) ;
9  sprintf ( comando , "%s EX4010-t.eps" , parametros .
       epstopdf ) ;
10 system ( comando ) ;
11 system ( "rm EX4010-t.eps" ) ;
12 system ( "rm EX4010z.dat" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  void Prepara_Histograma_Subtemas ( )
3  {
4  int i , maximo ;
5  FILE * Archivo_gnuplot , * Archivo_Datos ;
6  char Hilera_Antes [ 2000 ] , Hilera_Despues [ 2000 ] ;
7  char PG_command [ 1000 ] ;
8  gchar * materia ;
9  PGresult * res ;
10 char comando [ 2000 ] ;
11 materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
12 Archivo_Datos = fopen ( "EX4010y.dat" , "w" ) ;
13 for ( i = 0 ; i < N_subtemas ; i ++ )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  fprintf ( Archivo_Datos , "%d %Lf %Lf\n" , i , ( long
       double ) ( ( resumen_tema_subtema [ i ] . buenos +
       resumen_tema_subtema [ i ] . malos ) /
       N_estudiantes ) / N_preguntas * 100.0 ,
3  ( long double ) resumen_tema_subtema [ i ] . buenos / (
        resumen_tema_subtema [ i ] . buenos +
       resumen_tema_subtema [ i ] . malos ) * 100.0 ) ;
4  }
5  fclose ( Archivo_Datos ) ;
6  Archivo_gnuplot = fopen ( "EX4010-s.gp" , "w" ) ;
7  fprintf ( Archivo_gnuplot , "set term postscript eps
       enhanced color \"Times\" 12\n" ) ;
8  fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
        ) ;
9  fprintf ( Archivo_gnuplot , "set grid y2tics ytics\n" )
        ;
10 fprintf ( Archivo_gnuplot , "set output \"EX4010-s.eps
       \"\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set style fill solid border -1\n"
       ) ;
2  fprintf ( Archivo_gnuplot , "set boxwidth 0.8\n" ) ;
3  fprintf ( Archivo_gnuplot , "set xtics (" ) ;
4  for ( i = 0 ; i < N_subtemas ; i ++ )
5  {
6  sprintf ( PG_command , "SELECT descripcion_materia from
       BD_materias where codigo_materia = '%s' and
      codigo_tema = '%s' and codigo_subtema = '%s'" ,
7  materia , resumen_tema_subtema [ i ] . tema ,
      resumen_tema_subtema [ i ] . subtema ) ;
8  res = PQEXEC ( DATABASE , PG_command ) ;
9  sprintf ( Hilera_Antes , "\"%.44s (%3d)\" %d" ,
      PQgetvalue ( res , 0 , 0 ) , ( resumen_tema_subtema
       [ i ] . buenos + resumen_tema_subtema [ i ] .
      malos ) / N_estudiantes , i ) ;
10 hilera_GNUPLOT ( Hilera_Antes , Hilera_Despues ) ;
11 fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
12 if ( i == ( N_subtemas - 1 ) )
```

Código Preprocesado (Sin Pretty Print)

```
1
2  fprintf ( Archivo_gnuplot , ", " ) ;
3  PQclear ( res ) ;
4  }
5  fprintf ( Archivo_gnuplot , "set xtics rotate by -270
       scale 0\n" ) ;
6  fprintf ( Archivo_gnuplot , "set ytics rotate by 90\n"
       ) ;
7  fprintf ( Archivo_gnuplot , "set ytics 5,10\n" ) ;
8  fprintf ( Archivo_gnuplot , "set y2tics rotate by 90\n"
        ) ;
9  fprintf ( Archivo_gnuplot , "set y2tics 0,10\n" ) ;
10 fprintf ( Archivo_gnuplot , "set yrange [0:100.0]\n" )
       ;
11 fprintf ( Archivo_gnuplot , "set xlabel \" \"\n" ) ;
12 fprintf ( Archivo_gnuplot , "set size 1.15, 2.0\n" ) ;
13 fprintf ( Archivo_gnuplot , "set label 2 \"%% preguntas
       \" at graph 0.17, 0.86 left rotate by 90\n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set label 3 \"%% correctas\" at
       graph 0.22, 0.86 left rotate by 90\n" ) ;
2  fprintf ( Archivo_gnuplot , "plot \"EX4010y.dat\" using
       2 title \" \" lc 3 lt 1, \"\" using 3 title \" \"
       lc 2 lt 1\n" ) ;
3  fclose ( Archivo_gnuplot ) ;
4  sprintf ( comando , "%s EX4010-s.gp" , parametros .
       gnuplot ) ;
5  system ( comando ) ;
6  sprintf ( comando , "mv EX4010-s.gp %s" , parametros .
       ruta_gnuplot ) ;
7  system ( comando ) ;
8  sprintf ( comando , "%s EX4010-s.eps" , parametros .
       epstopdf ) ;
9  system ( comando ) ;
10 system ( "rm EX4010-s.eps" ) ;
11 system ( "rm EX4010y.dat" ) ;
12 g_free ( materia ) ;
13 }
```

Código Preprocesado (Sin Pretty Print)

```
1  double CDF ( long double X , long double Media , long
       double Desv )
2  {
3  return ( 0.5 * ( 1 + ( long double ) erf ( ( X - Media
       ) / ( Desv * M_SQRT2 ) ) ) ) ;
4  }
5  void Dificultad_vs_Discriminacion ( )
6  {
7  int i ;
8  FILE * Archivo_gnuplot ;
9  int Color , Smooth , Rotacion ;
10 int Total_preguntas ;
11 long double dificultad , discriminacion ;
12 int frecuencia ;
13 char comando [ 1000 ] ;
14 char
```

Código Preprocesado (Sin Pretty Print)

```
1  Hilera_Antes [ 2000 ] , Hilera_Despues [ 2000 ] ;
2  char * Colores [ ] = { "unset pm3d" ,
3  "set palette gray" ,
4  "set palette gray negative" ,
5  "set palette rgb 21,22,23" ,
6  "set palette rgb 34,35,36" ,
7  "set palette rgb 7,5,15" ,
8  "set palette rgb 3,11,6" ,
9  "set palette rgb 23,28,3" ,
10 "set palette rgb 33,13,10" ,
11 "set palette rgb 30,31,32" } ;
12 Color = ( int ) gtk_spin_button_get_value_as_int (
       SP_color ) ;
13 Rotacion = ( int ) gtk_spin_button_get_value_as_int (
       SP_rotacion ) ;
14 Smooth
```

Código Preprocesado (Sin Pretty Print)

```
1  = gtk_toggle_button_get_active ( CK_smooth ) ;
2  Calcular_Tabla ( ) ;
3  Archivo_gnuplot = fopen ( "EX4010-p.gp" , "w" ) ;
4  fprintf ( Archivo_gnuplot , "set term postscript eps
       enhanced color \"Times\" 12\n" ) ;
5  fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
       ) ;
6  fprintf ( Archivo_gnuplot , "set pm3d\n" ) ;
7  fprintf ( Archivo_gnuplot , "set style line 100 lt 5 lw
       0.5 lc 2\n" ) ;
8  fprintf ( Archivo_gnuplot , "set pm3d hidden3d 100\n" )
       ;
9  fprintf ( Archivo_gnuplot , "%s\n" , Colores [ Color ]
       ) ;
10 fprintf ( Archivo_gnuplot , "set size 1.55, 1.0\n" ) ;
11 if ( Niveles_Discriminacion < 12 )
12 fprintf ( Archivo_gnuplot , "set xtics -1.0, %4.3Lf,
       1.0 offset 1\n" , ( long double ) 2.0 / (
       Niveles_Discriminacion ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set xtics -1.0, %4.3Lf, 1.0 offset
       1\n" , 2 * ( long double ) 2.0 / (
       Niveles_Discriminacion ) ) ;
2  fprintf ( Archivo_gnuplot , "set ytics  0.0, %4.3Lf,
       1.0 offset 2\n" , ( long double ) 1.0 / (
       Niveles_Dificultad ) ) ;
3  fprintf ( Archivo_gnuplot , "set grid xtics\n" ) ;
4  fprintf ( Archivo_gnuplot , "set grid ytics\n" ) ;
5  fprintf ( Archivo_gnuplot , "set contour base\n" ) ;
6  fprintf ( Archivo_gnuplot , "set output \"EX4010p.eps
       \"\n" ) ;
7  hilera_GNUPLOT ( "set xlabel \"Discriminaci n\"\n" ,
       Hilera_Despues ) ;
8  fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
9  fprintf ( Archivo_gnuplot , "set ylabel \"Dificultad\"\
       n" ) ;
10 fprintf ( Archivo_gnuplot , "set xyplane at -0.5\n" ) ;
11 fprintf ( Archivo_gnuplot , "set hidden3d\n" ) ;
12 if ( Smooth ) fprintf ( Archivo_gnuplot , "set dgrid3 %
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "splot \"EX4010p.dat\" title \" \"
       with lines lt 1 lw 1 lc 2\n" ) ;
2  fclose ( Archivo_gnuplot ) ;
3  sprintf ( comando , "%s EX4010-p.gp" , parametros .
       gnuplot ) ;
4  system ( comando ) ;
5  sprintf ( comando , "mv EX4010-p.gp %s" , parametros .
       ruta_gnuplot ) ;
6  system ( comando ) ;
7  sprintf ( comando , "%s EX4010p.eps" , parametros .
       epstopdf ) ;
8  system ( comando ) ;
9  system ( "rm EX4010p.eps" ) ;
10 Archivo_gnuplot = fopen ( "EX4010c.gp" , "w" ) ;
11 fprintf ( Archivo_gnuplot , "set term postscript eps
       enhanced color \"Times\" 10\n" ) ;
12 fprintf ( Archivo_gnuplot , "set encoding iso_8859_1\n"
        ) ;
13 fprintf ( Archivo_gnuplot , "set pm3d\n" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "%s\n" , Colores [ Color ] ) ;
2  fprintf ( Archivo_gnuplot , "set size 1.55, 1.0\n" ) ;
3  fprintf ( Archivo_gnuplot , "set output \"EX4010c.eps
       \"\n" ) ;
4  hilera_GNUPLOT ( "set xlabel \"Discriminaci n\"\n" ,
       Hilera_Despues ) ;
5  fprintf ( Archivo_gnuplot , "%s" , Hilera_Despues ) ;
6  fprintf ( Archivo_gnuplot , "set ylabel \"Dificultad\"\
       n" ) ;
7  fprintf ( Archivo_gnuplot , "set view map\n" ) ;
8  fprintf ( Archivo_gnuplot , "set contour\n" ) ;
9  if ( Niveles_Discriminacion == 18 )
10 fprintf ( Archivo_gnuplot , "set xtics -1.0, %4.3Lf,
       1.0\n" , 2 * ( long double ) 2.0 / (
       Niveles_Discriminacion ) ) ;
11 else
12 fprintf ( Archivo_gnuplot , "set xtics -1.0, %4.3Lf,
       1.0\n" , ( long double ) 2.0 / (
       Niveles_Discriminacion ) ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_gnuplot , "set key at -1.1, -0.07 left box lt
       1\n" ) ;
2  if ( Smooth ) fprintf ( Archivo_gnuplot , "set dgrid3 %
       d,%d,gauss 0.15 0.15\n" , Niveles_Dificultad + 1 ,
       Niveles_Discriminacion + 1 ) ;
3  fprintf ( Archivo_gnuplot , "splot \"EX4010p.dat\"
       notitle with lines lt 1 lw 1 lc 2\n" ) ;
4  fclose ( Archivo_gnuplot ) ;
5  sprintf ( comando , "%s EX4010c.gp" , parametros .
       gnuplot ) ;
6  system ( comando ) ;
7  sprintf ( comando , "mv EX4010c.gp %s" , parametros .
       ruta_gnuplot ) ;
8  system ( comando ) ;
9  sprintf ( comando , "%s EX4010c.eps" , parametros .
       epstopdf ) ;
10 system ( comando ) ;
11 system ( "rm EX4010c.eps" ) ;
12 system ( "rm EX4010p.dat" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1  Calcular_Tabla ( )
2  {
3  int i , j , k ;
4  long double dificultad , discriminacion ;
5  FILE * Archivo_Datos ;
6  Niveles_Dificultad = ( int )
       gtk_spin_button_get_value_as_int ( SP_resolucion )
       ;
7  Niveles_Discriminacion = Niveles_Dificultad * 2 ;
8  for ( i = 0 ; i <= Niveles_Dificultad ; i ++ )
9  for ( j = 0 ; j <= Niveles_Discriminacion ; j ++ )
10 Frecuencia_total [ i ] [ j ] = 0 ;
11 for ( k = 0 ; k < N_preguntas ; k ++ )
12 {
13 dificultad = preguntas [ k ] . porcentaje ;
14 discriminacion
```

Código Preprocesado (Sin Pretty Print)

```
= preguntas [ k ] . Rpb ;
i = ( int ) round ( dificultad * Niveles_Dificultad ) ;
j = ( int ) round ( ( discriminacion + 1.0 ) / 2 *
    Niveles_Discriminacion ) ;
Frecuencia_total [ i ] [ j ] ++ ;
}
Archivo_Datos = fopen ( "EX4010p.dat" , "w" ) ;
for ( j = 0 ; j <= Niveles_Discriminacion ; j ++ )
{
for ( i = 0 ; i <= Niveles_Dificultad ; i ++ )
{
fprintf ( Archivo_Datos , "%Lf %Lf %d\n" , ( long
    double ) j / Niveles_Discriminacion * 2.0 - 1.0 , (
     long double ) i / Niveles_Dificultad ,
    Frecuencia_total [ i ] [ j ] ) ;
}
fprintf ( Archivo_Datos , "\n" ) ;
}
```

Código Preprocesado (Sin Pretty Print)

```
1
2  fclose ( Archivo_Datos ) ;
3  }
4  void Lista_de_Preguntas_Beamer ( FILE * Archivo_Latex ,
       GtkWidget * PB , long double base , long double
       limite )
5  {
6  int i , actual ;
7  char ejercicio_actual [ 7 ] = "00000" ;
8  char PG_command [ 3000 ] ;
9  PGresult * res , * res_tema ;
10 char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
11 char tema_actual [ CODIGO_TEMA_SIZE + 1 ] = "
       " ;
12 gchar * materia ;
13 char tema_descripcion [ 201 ] ;
14 char
```

Código Preprocesado (Sin Pretty Print)

```
1  opciones_frame [ 200 ] ;
2  int N_preguntas_ejercicio , N_validas ;
3  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
4  actual = 0 ;
5  while ( actual < N_preguntas )
6  {
7  if ( strcmp ( tema_actual , preguntas [ actual ] . tema
       ) != 0 )
8  {
9  strcpy ( tema_actual , preguntas [ actual ] . tema ) ;
10 sprintf ( PG_command , "SELECT descripcion_materia from
       bd_materias where codigo_materia = '%s' and
      codigo_tema = '%s' and codigo_subtema = '%s'" ,
11 materia , tema_actual , CODIGO_VACIO ) ;
12 res_tema = PQEXEC ( DATABASE , PG_command ) ;
13 strcpy ( tema_descripcion , PQgetvalue ( res_tema , 0 ,
       0 ) ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera_antes , "section{%s}" , tema_descripcion ) ;
2  hilera_LATEX ( hilera_antes , hilera_despues ) ;
3  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
4  PQclear ( res_tema ) ;
5  fprintf ( Archivo_Latex , "frame{tableofcontents[
       currentsection]}\n" ) ;
6  }
7  strcpy ( ejercicio_actual , preguntas [ actual ] .
       ejercicio ) ;
8  N_preguntas_ejercicio = N_validas = 0 ;
9  for ( i = actual ; ( i < N_preguntas ) && ( strcmp (
       ejercicio_actual , preguntas [ i ] . ejercicio ) ==
        0 ) ; i ++ )
10 {
11 if ( ! preguntas [ actual ] . excluir ) N_validas ++ ;
12 N_preguntas_ejercicio ++ ;
13 }
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( PG_command , "SELECT usa_header, texto_header from
       bd_texto_ejercicios, bd_ejercicios where
       codigo_ejercicio = '%s' and texto_ejercicio =
       consecutivo_texto" ,
2  ejercicio_actual ) ;
3  res = PQEXEC ( DATABASE , PG_command ) ;
4  if ( * PQgetvalue ( res , 0 , 0 ) == 't' )
5  {
6  if ( N_validas > 1 )
7  {
8  prepara_opciones ( opciones_frame , actual , - 1 ) ;
9  if ( N_validas == 2 )
10 {
11 fprintf ( Archivo_Latex , "begin{frame}%s{Preguntas %d
       y %d}\n" , opciones_frame , actual + 1 , actual + 2
        ) ;
12 fprintf ( Archivo_Latex , "textbf{Las preguntas %d y %d
       requieren la siguiente informaci'{o}n:}\n\n \n\n"
       , actual + 1 , actual + 2 ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  {
3  fprintf ( Archivo_Latex , "begin{frame}%s{Preguntas %d
       a %d}\n" , opciones_frame , actual + 1 , actual +
       N_preguntas_ejercicio ) ;
4  fprintf ( Archivo_Latex , "textbf{Las preguntas %d a %d
       requieren la siguiente informaci'{o}n:}\n\n \n\n"
       , actual + 1 , actual + N_preguntas_ejercicio ) ;
5  }
6  fprintf ( Archivo_Latex , "begin{beamercolorbox}[shadow
       =true , rounded=true]{pregunta}\n" ) ;
7  strcpy ( hilera_antes , PQgetvalue ( res , 0 , 1 ) ) ;
8  hilera_LATEX ( hilera_antes , hilera_despues ) ;
9  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
10 fprintf ( Archivo_Latex , "end{beamercolorbox}\n" ) ;
11 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
12 }
13 else
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  if ( ( N_validas > 0 ) && ! preguntas [ actual ] .
       excluir )
3  {
4  prepara_opciones ( opciones_frame , actual , -2 ) ;
5  fprintf ( Archivo_Latex , "{\n" ) ;
6  Marca_agua_ajuste ( Archivo_Latex , actual ) ;
7  sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d
       hfill {small %s}}" , opciones_frame , actual + 1 ,
       tema_descripcion ) ;
8  hilera_LATEX ( hilera_antes , hilera_despues ) ;
9  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
10 Imprime_pregunta_Beamer ( actual , Archivo_Latex ,
       PQgetvalue ( res , 0 , 1 ) , tema_descripcion ) ;
11 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
12 fprintf ( Archivo_Latex , "}\n" ) ;
13 }
14 N_preguntas_ejercicio
```

Código Preprocesado (Sin Pretty Print)

```
1  = 0 ;
2  actual ++ ;
3  }
4  }
5  PQclear ( res ) ;
6  if ( PB ) Update_PB ( PB , base + ( ( long double )
       actual / N_preguntas * limite ) ) ;
7  for ( i = 0 ; i < N_preguntas_ejercicio ; i ++ )
8  {
9  if ( ! preguntas [ actual + i ] . excluir )
10 {
11 fprintf ( Archivo_Latex , "{\n" ) ;
12 Marca_agua_ajuste ( Archivo_Latex , actual + i ) ;
13 if ( N_preguntas_ejercicio > 1 )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  prepara_opciones ( opciones_frame , actual + i , -2 )
       ;
3  sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d (%
       d - %d) hfill {small %s}}" ,
4  opciones_frame , actual + i + 1 , actual + 1 , actual +
        N_preguntas_ejercicio , tema_descripcion ) ;
5  hilera_LATEX ( hilera_antes , hilera_despues ) ;
6  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
7  }
8  else
9  {
10 prepara_opciones ( opciones_frame , actual + i , -2 )
       ;
11 sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d
       hfill {small %s}}" ,
12 opciones_frame , actual + i + 1 , tema_descripcion ) ;
13 hilera_LATEX ( hilera_antes , hilera_despues ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "%s\n" , hilera_despues ) ;
2  }
3  Imprime_pregunta_Beamer ( actual + i , Archivo_Latex ,
       " " , tema_descripcion ) ;
4  fprintf ( Archivo_Latex , "end{frame} \%\% CIERRA
       PREGUNTA\n" ) ;
5  fprintf ( Archivo_Latex , "}\n" ) ;
6  }
7  }
8  actual += N_preguntas_ejercicio ;
9  if ( PB ) Update_PB ( PB , base + ( ( long double )
       actual / N_preguntas * limite ) ) ;
10 }
11 g_free ( materia ) ;
12 }
13 void Marca_agua_ajuste ( FILE * Archivo_Latex , int i )
14 {
```

Código Preprocesado (Sin Pretty Print)

```
1
2  if ( preguntas [ i ] . revision_especial )
3  fprintf ( Archivo_Latex ,
4  "usebackgroundtemplate{vbox to paperheight{vfilhbox to
       paperwidth{hfilincludegraphics[width=0.5in]{.
       imagenes/fix.png}hfil}vfil}}\n" ) ;
5  }
6  void prepara_opciones ( char * opciones_frame , int i ,
       int k )
7  {
8  opciones_frame [ 0 ] = '\0' ;
9  if ( k == -2 )
10 {
11 if ( preguntas [ i ] . encoger && ! preguntas [ i ] .
       verbatim )
12 strcpy ( opciones_frame , "[shrink]" ) ;
13 else
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ i ] . encoger && preguntas [ i ] .
       verbatim )
2  strcpy ( opciones_frame , "[shrink, fragile]" ) ;
3  else
4  if ( ! preguntas [ i ] . encoger && preguntas [ i ] .
       verbatim )
5  strcpy ( opciones_frame , "[fragile]" ) ;
6  }
7  else
8  if ( k == - 1 )
9  {
10 if ( preguntas [ i ] . header_encoger && ! preguntas [
       i ] . header_verbatim )
11 strcpy ( opciones_frame , "[shrink]" ) ;
12 else
13 if ( preguntas [ i ] . header_encoger && preguntas [ i
       ] . header_verbatim )
14 strcpy
```

Código Preprocesado (Sin Pretty Print)

```
1  ( opciones_frame , "[shrink, fragile]" ) ;
2  else
3  if ( ! preguntas [ i ] . header_encoger && preguntas [
       i ] . header_verbatim )
4  strcpy ( opciones_frame , "[fragile]" ) ;
5  }
6  else
7  {
8  if ( preguntas [ i ] . encoger_opcion [ k ] && !
       preguntas [ i ] . verbatim_opcion [ k ] )
9  strcpy ( opciones_frame , "[shrink]" ) ;
10 else
11 if ( preguntas [ i ] . encoger_opcion [ k ] &&
       preguntas [ i ] . verbatim_opcion [ k ] )
12 strcpy ( opciones_frame , "[shrink, fragile]" ) ;
13 else
14 if
```

Código Preprocesado (Sin Pretty Print)

```
1  ( ! preguntas [ i ] . encoger_opcion [ k ] && preguntas
       [ i ] . verbatim_opcion [ k ] )
2  strcpy ( opciones_frame , "[fragile]" ) ;
3  }
4  }
5  void Imprime_pregunta_Beamer ( int i , FILE *
       Archivo_Latex , char * prefijo , char *
       tema_descripcion )
6  {
7  long double cota_inferior ;
8  char hilera_antes [ 10000 ] , hilera_despues [ 10000 ]
       ;
9  char PG_command [ 2000 ] ;
10 PGresult * res , * res_ejercicio ;
11 long double Por_A , Por_B , Por_C , Por_D , Por_E ,
       Total ;
12 gchar * materia ;
13 char opciones_frame [ 200 ] ;
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  j , N_flags ;
2  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
3  cota_inferior = 1.15 / 6.65 * 100.0 ;
4  sprintf ( PG_command , "SELECT texto_pregunta ,
       texto_opcion_A , texto_opcion_B , texto_opcion_C ,
       texto_opcion_D , texto_opcion_E from
       bd_texto_preguntas where codigo_unico_pregunta = '%
       s'" , preguntas [ i ] . pregunta ) ;
5  res = PQEXEC ( DATABASE , PG_command ) ;
6  fprintf ( Archivo_Latex , "begin{beamercolorbox}[shadow
       =true , rounded=true]{pregunta}\n" ) ;
7  sprintf ( hilera_antes , "%s %s" , prefijo , PQgetvalue
       ( res , 0 , 0 ) ) ;
8  hilera_LATEX ( hilera_antes , hilera_despues ) ;
9  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
10 fprintf ( Archivo_Latex , "end{beamercolorbox}\n" ) ;
11 Por_A = ( long double ) preguntas [ i ] .
       acumulado_opciones [ 0 ] / N_estudiantes * 100.0 ;
```

Código Preprocesado (Sin Pretty Print)

```
1  = ( long double ) preguntas [ i ] . acumulado_opciones
       [ 3 ] / N_estudiantes * 100.0 ;
2  Por_E = ( long double ) preguntas [ i ] .
       acumulado_opciones [ 4 ] / N_estudiantes * 100.0 ;
3  if ( preguntas [ i ] . slide [ 0 ] )
4  {
5  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
6  fprintf ( Archivo_Latex , "}\n" ) ;
7  prepara_opciones ( opciones_frame , i , 0 ) ;
8  fprintf ( Archivo_Latex , "{\n" ) ;
9  Marca_agua_ajuste ( Archivo_Latex , i ) ;
10 sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d -
       cont. hfill {small %s}}" , opciones_frame , i + 1 ,
        tema_descripcion ) ;
11 hilera_LATEX ( hilera_antes , hilera_despues ) ;
12 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
13 }
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{itemize}\n" ) ;
2  Imprime_Opcion_Beamer ( Archivo_Latex , res , Por_A , i
       , 0 ) ;
3  if ( preguntas [ i ] . slide [ 1 ] )
4  {
5  fprintf ( Archivo_Latex , "end{itemize}\n" ) ;
6  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
7  fprintf ( Archivo_Latex , "}\n" ) ;
8  prepara_opciones ( opciones_frame , i , 1 ) ;
9  fprintf ( Archivo_Latex , "{\n" ) ;
10 Marca_agua_ajuste ( Archivo_Latex , i ) ;
11 sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d -
       cont. hfill {small %s}}" , opciones_frame , i + 1 ,
        tema_descripcion ) ;
12 hilera_LATEX ( hilera_antes , hilera_despues ) ;
13 fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "begin{itemize}\n" ) ;
2  }
3  Imprime_Opcion_Beamer ( Archivo_Latex , res , Por_B , i
       , 1 ) ;
4  if ( preguntas [ i ] . slide [ 2 ] )
5  {
6  fprintf ( Archivo_Latex , "end{itemize}\n" ) ;
7  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
8  fprintf ( Archivo_Latex , "}\n" ) ;
9  prepara_opciones ( opciones_frame , i , 2 ) ;
10 fprintf ( Archivo_Latex , "{\n" ) ;
11 Marca_agua_ajuste ( Archivo_Latex , i ) ;
12 sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d -
       cont. hfill {small %s}}" , opciones_frame , i + 1 ,
        tema_descripcion ) ;
13 hilera_LATEX ( hilera_antes , hilera_despues ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "%s\n" , hilera_despues ) ;
2  fprintf ( Archivo_Latex , "begin{itemize}\n" ) ;
3  }
4  Imprime_Opcion_Beamer ( Archivo_Latex , res , Por_C , i
       , 2 ) ;
5  if ( preguntas [ i ] . slide [ 3 ] )
6  {
7  fprintf ( Archivo_Latex , "end{itemize}\n" ) ;
8  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
9  fprintf ( Archivo_Latex , "}\n" ) ;
10 prepara_opciones ( opciones_frame , i , 3 ) ;
11 fprintf ( Archivo_Latex , "{\n" ) ;
12 Marca_agua_ajuste ( Archivo_Latex , i ) ;
13 sprintf ( hilera_antes , "begin{frame}%s{Pregunta %d -
       cont. hfill {small %s}}" , opciones_frame , i + 1 ,
        tema_descripcion ) ;
14 hilera_LATEX
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera_antes , hilera_despues ) ;
2  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
3  fprintf ( Archivo_Latex , "begin{itemize}\n" ) ;
4  }
5  Imprime_Opcion_Beamer ( Archivo_Latex , res , Por_D , i
       , 3 ) ;
6  if ( preguntas [ i ] . slide [ 4 ] )
7  {
8  fprintf ( Archivo_Latex , "end{itemize}\n" ) ;
9  fprintf ( Archivo_Latex , "end{frame}\n" ) ;
10 fprintf ( Archivo_Latex , "}\n" ) ;
11 prepara_opciones ( opciones_frame , i , 4 ) ;
12 fprintf ( Archivo_Latex , "{\n" ) ;
13 Marca_agua_ajuste ( Archivo_Latex , i ) ;
14 sprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( hilera_antes , "begin{frame}%s{Pregunta %d - cont.
       hfill {small %s}}" , opciones_frame , i + 1 ,
       tema_descripcion ) ;
2  hilera_LATEX ( hilera_antes , hilera_despues ) ;
3  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
4  fprintf ( Archivo_Latex , "begin{itemize}\n" ) ;
5  }
6  Imprime_Opcion_Beamer ( Archivo_Latex , res , Por_E , i
       , 4 ) ;
7  fprintf ( Archivo_Latex , "end{itemize}\n" ) ;
8  PQclear ( res ) ;
9  N_flags = 0 ;
10 for ( j = 0 ; j <  16 ; j ++ ) N_flags += preguntas [ i
       ] . flags [ j ] ;
11 if ( N_flags && ! gtk_toggle_button_get_active (
       CK_sin_banderas ) )
12 {
13 fprintf ( Archivo_Latex , "end{frame}\n" ) ;
14 fprintf
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "}\n" ) ;
2  fprintf ( Archivo_Latex , "{\n" ) ;
3  sprintf ( hilera_antes , "begin{frame}{An lisis de
       Pregunta %d hfill {small %s}}" , i + 1 ,
       tema_descripcion ) ;
4  hilera_LATEX ( hilera_antes , hilera_despues ) ;
5  fprintf ( Archivo_Latex , "%s\n" , hilera_despues ) ;
6  Analiza_Ajuste ( Archivo_Latex , preguntas [ i ] , 1 )
       ;
7  Analiza_Banderas ( Archivo_Latex , preguntas [ i ] , 1
       , N_flags , i + 1 , tema_descripcion ) ;
8  }
9  g_free ( materia ) ;
10 }
11 void Imprime_Opcion_Beamer ( FILE * Archivo_Latex ,
       PGresult * res , long double Porcentaje , int
       pregunta , int opcion )
12 {
13 char hilera_antes [ 3000 ] , hilera_despues [ 3000 ] ;
```

Código Preprocesado (Sin Pretty Print)

```
1  ( preguntas [ pregunta ] . correcta == ( 'A' + opcion )
       )
2  fprintf ( Archivo_Latex , "item [correcta{%c}]" , 'A' +
       opcion ) ;
3  else
4  fprintf ( Archivo_Latex , "item [circled{%c}]" , 'A' +
       opcion ) ;
5  strcpy ( hilera_antes , PQgetvalue ( res , 0 , 1 +
       opcion ) ) ;
6  hilera_LATEX ( hilera_antes , hilera_despues ) ;
7  fprintf ( Archivo_Latex , "%s" , hilera_despues ) ;
8  fprintf ( Archivo_Latex , "\n\n{color{green} rule{%Lfcm
       }{5pt} {footnotesize textbf{texttt{%6.2Lf %%}}}} (
       textbf{%d}) hfill " ,
9  0.05 + 6.65 * Porcentaje / 100.0 , Porcentaje ,
10 preguntas [ pregunta ] . acumulado_opciones [ opcion ]
       ) ;
11 if ( preguntas [ pregunta ] . correcta == ( 'A' +
       opcion ) )
```

Código Preprocesado (Sin Pretty Print)

```
1  ( Archivo_Latex , "fcolorbox{black}{red}{color{white}
       $r_{pb}$ = textbf{%7.4Lf}}\n\n" , preguntas [
       pregunta ] . Rpb_opcion [ opcion ] ) ;
2  else
3  if ( preguntas [ pregunta ] . Rpb_opcion [ opcion ] >=
       0.3 )
4  fprintf ( Archivo_Latex , "fcolorbox{black}{blue}{color
       {white} $r_{pb}$ = textbf{%7.4Lf}}\n\n" , preguntas
        [ pregunta ] . Rpb_opcion [ opcion ] ) ;
5  else
6  if ( preguntas [ pregunta ] . Rpb_opcion [ opcion ] >=
       0.2 )
7  fprintf ( Archivo_Latex , "fcolorbox{black}{cyan}{color
       {white} $r_{pb}$ = textbf{%7.4Lf}}\n\n" , preguntas
        [ pregunta ] . Rpb_opcion [ opcion ] ) ;
8  else
9  if ( preguntas [ pregunta ] . acumulado_opciones [
       opcion ] == 0 )
10 fprintf ( Archivo_Latex , "fcolorbox{black}{red}{color{
```

Código Preprocesado (Sin Pretty Print)

```
1
2  fprintf ( Archivo_Latex , "fcolorbox{black}{white}{
       color{black} $r_{pb}$ = textbf{%7.4Lf}}\n\n" ,
       preguntas [ pregunta ] . Rpb_opcion [ opcion ] ) ;
3  fprintf ( Archivo_Latex , "setlength{fboxrule}{2
       fboxrule}\n" ) ;
4  fprintf ( Archivo_Latex , "\n\nfcolorbox{black}{red}{
       color{white} $starstarstar$ textbf{CORRECTA}
       $starstarstar$}" ) ;
5  fprintf ( Archivo_Latex , "setlength{fboxrule}{0.5
       fboxrule}\n" ) ;
6  }
7  else
8  {
9  if ( preguntas [ pregunta ] . Rpb_opcion [ opcion ] >
       0.3 )
10 fprintf ( Archivo_Latex , "fcolorbox{black}{red}{color{
       white} $r_{pb}$ = textbf{%7.4Lf}}\n\n" , preguntas
       [ pregunta ] . Rpb_opcion [ opcion ] ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1
2  if ( preguntas [ pregunta ] . Rpb_opcion [ opcion ] <=
       - 0.3 )
3  fprintf ( Archivo_Latex , "fcolorbox{black}{blue}{color
       {white} $r_{pb}$ = textbf{%7.4Lf}}\n\n" , preguntas
        [ pregunta ] . Rpb_opcion [ opcion ] ) ;
4  else
5  fprintf ( Archivo_Latex , "fcolorbox{black}{white}{
       color{black} $r_{pb}$ = textbf{%7.4Lf}}\n\n" ,
       preguntas [ pregunta ] . Rpb_opcion [ opcion ] ) ;
6  }
7  fprintf ( Archivo_Latex , "\n\n" ) ;
8  }
9  void Busca_pregunta_mala_Beamer ( int N )
10 {
11 int i ;
12 int Exclusiones_previas [ N ] ;
13 int low , high , mid ;
14 int
```

Código Preprocesado (Sin Pretty Print)

```
1  resultado ;
2  char mensaje_error [ 2000 ] ;
3  gtk_widget_show ( window6 ) ;
4  Update_PB ( PB_revisando_beamer , 0.0 ) ;
5  for ( i = 0 ; i < N ; i ++ ) Exclusiones_previas [ i ]
       = preguntas [ i ] . excluir ;
6  low = 0 ;
7  high = N - 1 ;
8  while ( high != low )
9  {
10 mid = ( high + low ) / 2 ;
11 for ( i = 0 ; i < N ; i ++ ) preguntas [ i ] . excluir
       = 1 ;
12 for ( i = low ; i <= mid ; i ++ ) preguntas [ i ] .
       excluir = 0 ;
13 for ( i = low ; i <= mid ; i ++ ) preguntas [ i ] .
       excluir = Exclusiones_previas [ i ] ;
14 resultado
```

Código Preprocesado (Sin Pretty Print)

```
1  = Genera_Beamer_reducido ( ) ;
2  Update_PB ( PB_revisando_beamer , ( long double ) (
       N_preguntas - ( high - mid ) ) / N_preguntas ) ;
3  if ( resultado )
4  {
5  low = mid + 1 ;
6  }
7  else
8  {
9  high = mid ;
10 }
11 }
12 sprintf ( mensaje_error , "La pregunta n mero <b>%2d</
       b> de este examen tiene\nalg n conflicto con <b><i
       >Beamer</i></b>. Esta corresponde\na la pregunta <b
       >%s.%d</b> de la Base de Datos.\n\nRevise
       detalladamente la sintaxis de\nesta pregunta.
       Considere marcar ya\nsea la casilla \"<b>verbatim</
       b>\" o la casilla\n\"<b>Excluir</b>\" en la
```

Código Preprocesado (Sin Pretty Print)

```
1  ( LB_error_encontrado_Beamer , mensaje_error ) ;
2  for ( i = 0 ; i < N ; i ++ ) preguntas [ i ] . excluir
       = Exclusiones_previas [ i ] ;
3  gtk_widget_hide ( window6 ) ;
4  gtk_widget_show ( window7 ) ;
5  }
6  int Genera_Beamer_reducido ( )
7  {
8  int k ;
9  char codigo [ 10 ] ;
10 int resultado_OK = 0 ;
11 FILE * Archivo_Latex ;
12 gchar * size , * font , * color , * estilo , * materia
       , * fecha ;
13 int aspecto ;
14 char
```

Código Preprocesado (Sin Pretty Print)

```
1  Directorio [ 3000 ] ;
2  Banderas [ 0 ] = Banderas [ 1 ] = Banderas [ 2 ] =
       Banderas [ 3 ] = Banderas [ 4 ] = Banderas [ 5 ] =
       0 ;
3  k = ( int ) gtk_spin_button_get_value_as_int (
       SP_examen ) ;
4  sprintf ( codigo , "%05d" , k ) ;
5  materia = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_materia ) , 0 , -1 ) ;
6  fecha = gtk_editable_get_chars ( GTK_EDITABLE (
       EN_fecha ) , 0 , -1 ) ;
7  estilo = gtk_combo_box_get_active_text ( CB_estilo ) ;
8  color = gtk_combo_box_get_active_text ( CB_color ) ;
9  font = gtk_combo_box_get_active_text ( CB_font ) ;
10 size = gtk_combo_box_get_active_text ( CB_size ) ;
11 aspecto = gtk_combo_box_get_active ( CB_aspecto ) ;
12 Establece_Directorio ( Directorio , materia , fecha + 6
       , fecha + 3 , fecha ) ;
13 Archivo_Latex = fopen ( "analisis-beamer.tex" , "w" ) ;
```

Código Preprocesado (Sin Pretty Print)

```
1 ( Archivo_Latex , aspecto , size , estilo , color ,
      font ) ;
2 Beamer_Preguntas ( Archivo_Latex , NULL , 0.0 , 0.0 ) ;
3 Beamer_Cierre ( Archivo_Latex ) ;
4 fclose ( Archivo_Latex ) ;
5 resultado_OK = latex_2_pdf ( & parametros , Directorio
      , parametros . ruta_latex , "analisis-beamer" , 0 ,
       NULL , 0.0 , 0.0 , NULL , NULL ) ;
```