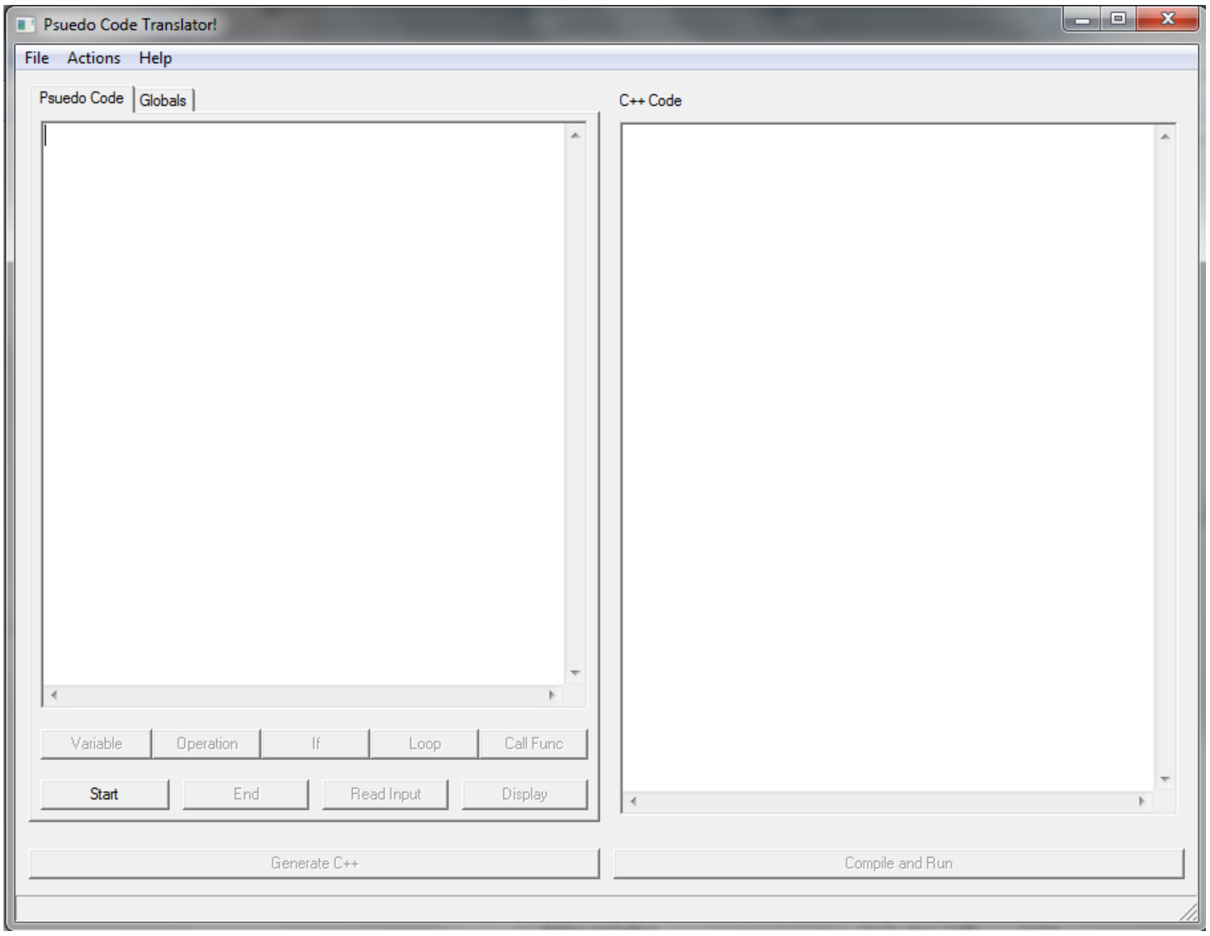
Pseudo Code Translator



Computer Science Project 2013-14

<u>Developed By</u>:
Arjun Rao - 12 A
Sriram Desai - 12 B

Acknowledgement

We would like to express deep gratitude to our principal, Mrs. Deepa Sridhar for providing us with the opportunity to work on this project. We would also like to thank our teachers, Mrs. Kavitha Purushotham, Mrs. Smitha Ravindran, Mrs. Kanchana Pravin Kumar, for their patient guidance, enthusiastic encouragement and useful critiques of this project work.

We also extend our thanks to Mr. Livin, for his help in setting up the resources required for the project. Lastly we would like to thank our parents and friends for supporting us throughout the course of this project.

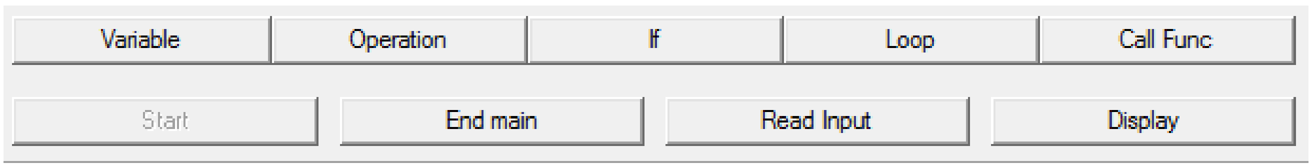
Contents

Synopsis

Our project, Pseudo Code Translator, basically allows the user to write, edit and save pseudo code and also translates it into the equivalent C++ code. This application is meant for learning to program, and to help simplify understanding C++. As soon as the application is opened the user sees a split screen, on the left is the pseudo code part and the right is the C++ part. The pseudo code part has 2 tabs, one is the pseudo code and the other is the globals tab.

The pseudo code tab is meant for building the functions, both main() as well as user defined functions. The global tab is for all global declarations, from global variables to classes!

Since pseudo code structure varies from person to person, and depends on the user's discretion, we have agreed upon certain syntax for pseudo code, and to help understand this, we have provided for quick insert buttons on the program:



These buttons insert the corresponding code snippets, in pseudo code, hence it is easier for the learner to code, as all he needs to do is figure out the logic for the program, and click on the buttons. The buttons, further help in understanding the syntax, which is for the most part, general English statements.

For example,

To declare a variable, the syntax is as follows,

Variable sum as int

Or to write a function,

Function sum parameters: int a,int b returns int

The program, then converts the pseudo code to its equivalent C++ code, at the click of a button. We also add in the required include statements for a basic program to work, all though we do provide an option to include additional header files, if required.

System Requirements

For running the application –

1. OS - Windows 2000 and above
2. 34 bit CPU
3. 128MB RAM
4. 12+ MB hard disk space

For viewing and editing source code –

1. OS – Windows 7 and above
2. 32 bit CPU
3. wxWidgets library for C++
4. Any IDE which supports wxWidgets (eg: Orwell DevCpp 5.2.0.1)
5. Optional: wxFromBuilder

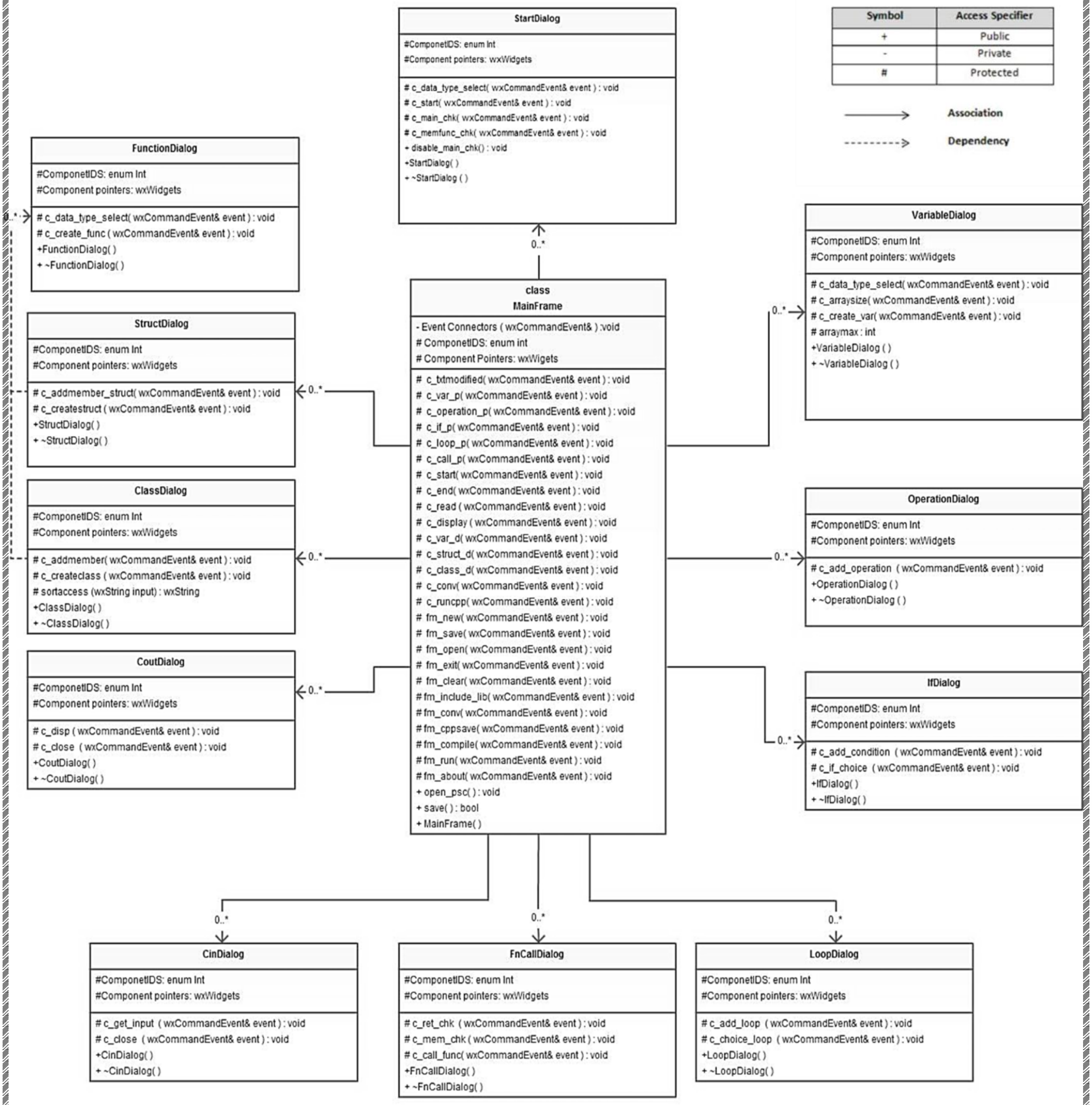
Need for the Pseudo Code Translator

As mentioned in the synopsis this application is meant for people learning to program or those learning to write C++. We found that are very few applications (if any), have been developed that help learn a language through pseudo code, of course there are softwares like scratch, which make learning to program easier, but they do not use C++. We found that sometimes, for those who are first time programmers, learning C++ or C, becomes daunting due to the strictly typed nature of these languages, and they end up focusing a lot on learning the syntax, than learning to program, or write logical instructions. To address this issue, we decided to write a program that helps one learn to program, and at the same time, highlight the syntaxes of the language.

UML CLASS DIAGRAM

UML SYMBOLS

Symbol	Access Specifier
+	Public
-	Private
#	Protected



Association
Dependency
StartDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_data_type_select(wxCommandEvent& event) : void
# c_start(wxCommandEvent& event) : void
# c_main_chk(wxCommandEvent& event) : void
# c_memfunc_chk(wxCommandEvent& event) : void
+ disable_main_chk() : void
+StartDialog( )
+ ~StartDialog ( )
0..*
FunctionDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_data_type_select(wxCommandEvent& event) : void
# c_create_func (wxCommandEvent& event) : void
+FunctionDialog( )
+ ~FunctionDialog( )
0..*
StructDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_addmember_struct(wxCommandEvent& event) : void
# c_createstruct (wxCommandEvent& event) : void
+StructDialog( )
+ ~StructDialog( )
0..*
ClassDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_addmember(wxCommandEvent& event) : void
# c_createclass (wxCommandEvent& event) : void
# sortaccess (wxString input) : wxString
+ClassDialog( )
+ ~ClassDialog( )
0..*
CoutDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_disp (wxCommandEvent& event) : void
# c_close (wxCommandEvent& event) : void
+CoutDialog( )
+ ~CoutDialog( )
0..*
class
MainFrame
- Event Connectors (wxCommandEvent&) :void
# ComponetIDS: enum int
# Component Pointers: wxWigets
# c_txtmodified(wxCommandEvent& event) : void
# c_var_p(wxCommandEvent& event) : void
# c_operation_p(wxCommandEvent& event) : void
# c_if_p(wxCommandEvent& event) : void
# c_loop_p(wxCommandEvent& event) : void
# c_call_p(wxCommandEvent& event) : void
# c_start(wxCommandEvent& event) : void
# c_end(wxCommandEvent& event) : void
# c_read (wxCommandEvent& event) : void
# c_display (wxCommandEvent& event) : void
# c_var_d(wxCommandEvent& event) : void
# c_struct_d(wxCommandEvent& event) : void
# c_class_d(wxCommandEvent& event) : void
# c_conv(wxCommandEvent& event) : void
# c_runcpp(wxCommandEvent& event) : void
# fm_new(wxCommandEvent& event) : void
# fm_save(wxCommandEvent& event) : void
# fm_open(wxCommandEvent& event) : void
# fm_exit(wxCommandEvent& event) : void
# fm_clear(wxCommandEvent& event) : void
# fm_include_lib(wxCommandEvent& event) : void
# fm_conv(wxCommandEvent& event) : void
# fm_cppsave(wxCommandEvent& event) : void
# fm_compile(wxCommandEvent& event) : void
# fm_run(wxCommandEvent& event) : void
# fm_about(wxCommandEvent& event) : void
+ open_psc( ) : void
+ save( ) : bool
+ MainFrame( )
0..*
VariableDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_data_type_select(wxCommandEvent& event) : void
# c_arraysize(wxCommandEvent& event) : void
# c_create_var(wxCommandEvent& event) : void
# arraymax : int
+VariableDialog ( )
+ ~VariableDialog ( )
0..*
OperationDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_add_operation (wxCommandEvent& event) : void
+OperationDialog ( )
+ ~OperationDialog ( )
0..*
IfDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_add_condition (wxCommandEvent& event) : void
# c_if_choice (wxCommandEvent& event) : void
+IfDialog( )
+ ~IfDialog( )
0..*
CinDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_get_input (wxCommandEvent& event) : void
# c_close (wxCommandEvent& event) : void
+CinDialog( )
+ ~CinDialog( )
0..*
FnCallDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_ret_chk (wxCommandEvent& event) : void
# c_mem_chk (wxCommandEvent& event) : void
# c_call_func(wxCommandEvent& event) : void
+FnCallDialog( )
+ ~FnCallDialog( )
0..*
LoopDialog
#ComponetIDS: enum Int
#Component pointers: wxWidgets
# c_add_loop (wxCommandEvent& event) : void
# c_choice_loop (wxCommandEvent& event) : void
+LoopDialog( )
+ ~LoopDialog( )

Header Files

WxWidgets Library header files	User defined header files -
utils.h	class_dialog.h
artprov.h	dialog_if.h
string.h	dialog_loop.h
textctrl.h	dialog_operation.h
gdicmn.h	frame_var.h
font.h	struct_dialog.h
colour.h	dialog_cin.h
settings.h	dialog_cout.h
button.h	start_dialog.h
sizer.h	call_dialog.h
panel.h	control.h
bitmap.h	dialog_include.h
image.h	
notebook.h	
stattext.h	
menu.h	
statusbr.	
frame.h	
msgdlg.h	
intl.h	
filedlg.h	
textfile.h	

Function Description

MainFrame :: MainFrame()

The constructor, responsible for creating the main UI window, and handling initialization of global variables, required for execution. The MainFrame Dialog is the central core component of the entire program. All UI events are directly or indirectly handled by the MainFrame class.

c_txtmodified(wxCommandEvent& event)

This function checks, if the user has modified any text in the pseudo code tab, and declares the state of the file as not saved, and hence prompts the user to save the file while converting to pseudo code.

c_var_p(wxCommandEvent& event)

The function is triggered when the user clicks the variable quick insert button. It opens the create variable dialog, with properties required for variable declaration.

c_operation_p(wxCommandEvent& event)

The function is triggered when the user clicks the operation quick insert button. It opens text input dialog, where the user can type in the required operation.

Eg sum = a + b

c_if_p(wxCommandEvent& event)

The function is triggered when the user clicks the 'if' quick insert button. It opens dialog a dialog for inserting condition statements. It also adds an indent to the pseudo code.

c_loop_p(wxCommandEvent& event)

The function is triggered when the user clicks the 'loop' quick insert button. It opens dialog a dialog for inserting loop blocks statements. It also adds an indent to the pseudo code.

c_call_p(wxCommandEvent& event)

The function is triggered when the user clicks the 'call function' quick insert button. It opens dialog a dialog for inserting a function call.

c_start(wxCommandEvent& event)

The function is triggered when the user clicks the 'start' quick insert button. It is opens a dialog to begin a function definition. The start dialog is used to define the main function, non-member, and member functions.

c_end(wxCommandEvent& event)

The function is triggered when the user clicks the 'End ____ ' quick insert button. Label of this button changes with respect to the block that is being defined. It is used to end the current block and un indent by one step.

The label may be 'End If' if the user is currently defining an If block, or it may be 'End for' or 'End while' if the user is defining a looping construct.

c_read (wxCommandEvent& event)

The function is triggered when the user clicks the 'Read Input' quick insert button. It opens dialog a dialog for inserting input statements.

c_display (wxCommandEvent& event)

The function is triggered when the user clicks the 'Display' quick insert button. It opens dialog a dialog for inserting output statements.

c_conv(wxCommandEvent& event)

The function is triggered when the user clicks the 'Generate C++' quick insert button or the Generate C++ file option. It intern calls other functions to convert the pseudo code to C++ code.

Source Code

main.cpp

```
#include <wx/wx.h>
#include "mainframe.h"

class Pct : public wxApp
{
      public:
      virtual bool OnInit();
};

IMPLEMENT_APP(Pct)

bool Pct::OnInit()
{

    MainFrame *frame = new MainFrame(NULL);
    if(argc==2)
      {
            CurrentDocPath=argv[1];
            frame->open_psc();
      }
    frame->Show(TRUE);
    SetTopWindow(frame);
    return TRUE;
}
```

MainFrame.h

```
#ifndef __MAINFRAME_H__
#define __MAINFRAME_H__


#include <wx/utils.h>
#include <wx/artprov.h>
#include <wx/string.h>
#include <wx/textctrl.h>
#include <wx/gdicmn.h>
#include <wx/font.h>
#include <wx/colour.h>
#include <wx/settings.h>
#include <wx/button.h>
#include <wx/sizer.h>
#include <wx/panel.h>
#include <wx/bitmap.h>
#include <wx/image.h>
#include <wx/icon.h>
#include <wx/notebook.h>
#include <wx/stattext.h>
#include <wx/menu.h>
#include <wx/statusbr.h>
#include <wx/frame.h>
#include <wx/msgdlg.h> // for message box dialog
#include <wx/filedlg.h>//for file dialog
#include <wx/intl.h> // for _(" some text ") macro
#include <wx/textfile.h>
```

```
#include "class_dialog.h"
#include "dialog_if.h"
#include "dialog_loop.h"
#include "dialog_operation.h"
#include "frame_var.h"
#include "struct_dialog.h"
#include "dialog_cin.h"
#include "dialog_cout.h"
#include "start_dialog.h"
#include "call_dialog.h"
#include "control.h"
#include "dialog_include.h"



class MainFrame : public wxFrame
{
      DECLARE_EVENT_TABLE()

      public:
            enum
            {
                  wxID_code_input = 1000,
                  wxID_var_p,
                  wxID_operation_p,
                  wxID_if_p,
                  wxID_loop_p,
                  wxID_call_p,
                  wxID_func_p,
                  wxID_start,
                  wxID_end,
                  wxID_read,
                  wxID_display,
                  wxID_data_input,
                  wxID_var_d,
                  wxID_struct_d,
                  wxID_class_d,
                  wxID_cppOP,
                  wxID_conv,
                  wxID_runcpp,
                  wxID_new,
                  wxID_save,
                  wxID_save_as,
                  wxID_open,
                  wxID_exit,
                  wxID_clear,
                  wxID_generate,
                  wxID_convert,
                  wxID_compile,
                  wxID_run,
                  wxID_include_lib,
                  wxID_about
            };

            wxPanel* mainpanel;
            wxPanel* m_Editors;
            wxNotebook* m_pseudo_code;
            wxPanel* m_psuedo_panel;
            wxTextCtrl* t_code_input;
            wxPanel* m_pseudo_buttons;
            wxPanel* m_cls_struct_panel;
            wxTextCtrl* t_data_input;
            wxPanel* m_buttons;
            wxPanel* m_cpp_code;
            wxStaticText* t_cpp;
            wxTextCtrl* t_cpp_output;
            wxPanel* m_UtilityBar;
            wxButton* m_conv;
            wxMenuBar* m_menubar;
            wxMenu* m_Filemenu;
            wxMenu* m_ActionMenu;
            wxMenu* m_HelpMenu;
            wxStatusBar* m_statusBar;

            void c_txtmodified( wxCommandEvent& event );
            void c_var_p( wxCommandEvent& event );
            void c_operation_p( wxCommandEvent& event );
            void c_if_p( wxCommandEvent& event );
            void c_loop_p( wxCommandEvent& event );
            void c_call_p( wxCommandEvent& event );
            void c_start( wxCommandEvent& event );
            void c_end( wxCommandEvent& event );
            void c_read ( wxCommandEvent& event );
            void c_display ( wxCommandEvent& event );
            void c_var_d( wxCommandEvent& event );
            void c_struct_d( wxCommandEvent& event );
            void c_class_d( wxCommandEvent& event );
            void c_conv( wxCommandEvent& event );
            void c_runcpp( wxCommandEvent& event );
            void fm_new( wxCommandEvent& event );
            void fm_save( wxCommandEvent& event );
            void fm_open( wxCommandEvent& event );
            void fm_exit( wxCommandEvent& event );
            void fm_clear( wxCommandEvent& event );
            void fm_include_lib( wxCommandEvent& event );
            void fm_conv( wxCommandEvent& event );
            void fm_cppsave( wxCommandEvent& event );
            void fm_compile( wxCommandEvent& event );
            void fm_run( wxCommandEvent& event );
            void fm_about( wxCommandEvent& event );


      public:
            void open_psc();
            bool save();
            MainFrame( wxWindow* parent, wxWindowID id = wxID_ANY, const
wxString& title = wxT("Psuedo Code Translator!"), const wxPoint& pos =
wxDefaultPosition, const wxSize& size = wxSize( 906,698 ), long style =
wxDEFAULT_FRAME_STYLE|wxMAXIMIZE|wxTAB_TRAVERSAL );
};

#endif
```

Control.h

```
#ifndef __CONTROL_H__
#define __CONTROL_H__
#include <wx/artprov.h>
#include <wx/wx.h>
#include <wx/string.h>
#define TABS 10
#include "mainframe.h"


extern wxString insert_code;
extern int n_ind;//extent of indentation
extern long int stackcount;
extern bool is_main_def;//to check if main function is defined
extern wxString endlabel_text;//To hold the label for end button;
extern wxString include_lib_list[100];//to hold the include liberaries list
extern long int n_lib;//number of liberaries included...
extern bool is_file_saved;//true is it is saved
extern bool is_cpp_saved;
extern wxString CurrentDocPath; //for saving the psc file.
extern wxString CurrentDocPath2;//for saving cpp file
extern wxString cppinclude,cppglobal,cppcode;

class stk_present_block//for changing the button labels from child dialog,
{
            struct node
            {
                  wxString text;
                  node *link;
            }*top;//text will be going into the label of the end button
      public:
            stk_present_block()
            {
                  top=NULL;
            }
            void push(wxString input);
            wxString pop();
            wxString getlist(); //returns all elemets of stack as a string
            wxString getTop()
            {
                  return top->text;
            }
            bool IsEmpty()
            {
                  if(top==NULL)
                        return true;
                  else
                        return false;
            }
};
extern stk_present_block stk_p_block;
class stk_line_number//for numbering the lines in pseudo cade window
{
      int l_no[100],top;
      public:
            stk_line_number()
            {
                  top=-1;
            }
            void push(int n);
            int pop();
            void inc_step()//increments l_no[top]
            {
                  l_no[top]++;
            }
            int ret_top()
            {
                  return top;
            }
            void ret_array(int x[])
            {
                  for(int i=0;i<=top;i++)
                  x[i]=l_no[i];
            }
            void reset_top()
            {
                  top = -1;
            }
};
extern stk_line_number stk_l_no;
extern int temp_l_no[100];


//conversion functions
wxString conv_variable(wxString input);
wxString conv_g_func(wxString input);
wxString conv_class(wxString input);
wxString conv_struct(wxString input);
wxString conv_globals();
wxString conv_include_lib();
wxString get_data_in_file();
wxString conv_p_func(wxString input);
wxString conv_p_callfn(wxString input);
wxString conv_io(wxString input);
wxString conv_psuedo();
wxString conv_p_elseif(wxString input);
wxString conv_p_if(wxString input);
wxString conv_p_forloop(wxString input);
wxString conv_p_whileloop(wxString input);



#endif //__CONTROL_H__
```

`Control.cpp` – Has all functions which convert the pseudo code to C++ code.

```
#include "control.h"

wxString insert_code;
wxString endlabel_text;
wxString CurrentDocPath;
wxString CurrentDocPath2;
wxString include_lib_list[100];
int n_ind;
long int n_lib;
long int stackcount;
bool is_main_def;
bool is_file_saved;
bool is_cpp_saved;
wxString cppinclude,cppglobal,cppcode;
```

```
void stk_present_block :: push(wxString input)
{
      node *temp=new node;
      temp->link=NULL;
      temp->text<<input;
      if(top==NULL)
      {
            top=temp;
      }
      else
      {
            temp->link=top;
            top=temp;
      }
}
wxString stk_present_block :: pop()
{
      node *temp;
      wxString x;
      if(top!=NULL){
            temp=top;
            x<<temp->text;
            top=top->link;
            temp->link=NULL;
            delete temp;
            return x;
      }
      else
            return _T("Stack Empty");
}
wxString stk_present_block :: getlist()
{
      stackcount = 0;
      stk_present_block templist;
      wxString return_string;
      return_string.Empty();
      while(!this->IsEmpty())
      {
            stackcount++;
            return_string<<this->getTop()<<"%";
            templist.push(this->pop());
      }
      while(!templist.IsEmpty())
      {
            this->push(templist.pop());
      }
      return return_string;
}
void stk_line_number :: push(int n)
{
      top++;
      l_no[top]=n;
}
int stk_line_number :: pop()
{
      top--;
      return l_no[top+1];
}

stk_line_number stk_l_no;
```

```
int temp_l_no[100];
stk_present_block stk_p_block;

wxString get_data_in_file()//returns the contents of the the .psc file
{
      wxTextFile file;
      wxString tempString;
      file.Open(CurrentDocPath);
      if(file.IsOpened())
    {
         tempString.Empty();
             //store file data for further processing
             for (wxString str = file.GetFirstLine(); !file.Eof();str =
file.GetNextLine())
             {
                  tempString += str;
                  tempString += "\n";
             }
      }
      return tempString;
}

wxString conv_include_lib()
{
      wxString tempString;
      tempString.Empty();
      tempString=get_data_in_file();
      tempString.Replace("@Begin\n","");
      tempString=tempString.AfterFirst('^');
      tempString=tempString.BeforeFirst('$');
      tempString.Replace("%","");
      if(tempString.Contains(wxT("iostream")))
      tempString<<"\nusing namespace std;\n";
      return tempString;
}

wxString conv_p_func(wxString input)
{
      wxString output,fname,rtype,params;
      input = input.AfterFirst(' ');
      fname = input.BeforeFirst(' ');
      input = input.AfterFirst(' ');
      input = input.AfterFirst(' ');
      input.Replace("returns","%returns");
      params = input.BeforeFirst('%');
      input = input.AfterFirst('%');
      rtype = input.AfterFirst(' ');
      output.Empty();
      if(params.Contains(wxT("none"))) params.Empty();
      if (fname == "main") rtype = "int";
      output<< rtype <<" " << fname << "(" << params <<")\n{\n";
      return output;
}

wxString conv_p_elseif(wxString input)//input =  else if condition
{
      wxString output,condition;
      input = input.AfterFirst(' ');
      condition = input.AfterFirst(' ');
      output.Empty();
      output<<"else if( " << condition << " )\n";
```

```
		return output;
	}

	wxString conv_p_if(wxString input)//input = if condition
	{
		wxString output,condition;
		condition = input.AfterFirst(' ');
		output.Empty();
		output<<"if( " << condition << " )\n";
		return output;
	}

	wxString conv_p_forloop(wxString input)//input =
	{
		wxString output,initial,condition,reinitial;
		input.Replace("for loop from","for");
		initial = input.AfterFirst(' ');
		input = initial;//i=0 until i<5 update i=i+2
		initial = initial.BeforeFirst(' ');
		input = input.AfterFirst(' ');//until i<5 update i=i+2
		input = input.AfterFirst(' ');//i<5 update i=i+2
		condition = input.BeforeFirst(' ');
		input = input.AfterFirst(' ');//update i=i+2
		reinitial = input.AfterFirst(' ');
		output.Empty();
		output<<"for("<<initial<<";"<<condition<<";"<<reinitial<<")\n";
		return output;
	}


	wxString conv_p_whileloop(wxString input)//input =  while loop until x>asd
	{
		wxString output,condition;
		input.Replace("while loop until ","while ");
		condition = input.AfterFirst(' ');
		output.Empty();
		output<<"while( "<<condition <<" )\n"; //output =  while( i<5 )\n
		return output;
	}
	wxString conv_p_callfn(wxString input)//Call sample.asdf() store returned
	value in abc
	{
		wxString output,fname,rvar;
		fname = input.AfterFirst(' ');
		fname = fname.BeforeFirst(' '); //has sample.asdf()
		rvar = input.AfterLast(' '); //abc
		output.Empty();
		if(input.Contains(wxT("returned")))
		{
			rvar<<" = ";
			output<<rvar<<fname<<";";
		}
		else
		{
			output<<fname<<";";
		}
		return output;
	}
```

```
wxString conv_io(wxString input)//To convert to CIN and COUT statements
{
      wxString output;
      output.Empty();
      if(input.Contains(wxT("Read")))
      {
            output<<"cin>>";
            input=input.AfterFirst('r');
            output<<input<<";";
      }
      else if(input.Contains(wxT("Display")))
      {
            output<<"cout<<";
            input=input.AfterFirst('y');
            output<<input<<";";
      }
      return output;
}

wxString conv_psuedo()//Parse psuedo code Tab
{
      wxString tempString,tstring2,convString,pString;
      int indcount=0;//counting indents of normal statements
      int endindcount = 0;//counting indent of end statements
      tempString.Empty();
      tempString=get_data_in_file();
      convString.Empty();
      tempString.Replace("@Begin\n","");
      pString= tempString.BeforeFirst ('$');
      //the main parser:
      pString.Replace("\n","$\n");
      tempString.Empty();
      while(pString.Contains(wxT("$")))
      {
            tempString.Empty();
          tempString = pString.BeforeFirst('$');//tempstring stores a line
          //check if its a function statement:
          if(tempString.Contains("Display") &&
(!tempString.Matches("Function*")))
          {
                tstring2 = tempString.BeforeFirst('D');
                indcount = tstring2.Freq('.');
                for(int i=0; i<indcount;i++)convString<<"\t";
                tempString.Replace(". Display ",". %Display ");
                tempString = tempString.AfterFirst('%');
                convString << conv_io(tempString)<<"\n";
            }
            else if(tempString.Contains("Read"))
          {
                tstring2 = tempString.BeforeFirst('R');
                indcount = tstring2.Freq('.');
                for(int i=0; i<indcount;i++)convString<<"\t";
                tempString.Replace("Read ","%Read ");
                tempString = tempString.AfterFirst('%');
                convString << conv_io(tempString)<<"\n";
            }
            else if(tempString.Contains("Call "))
          {
                tstring2 = tempString.BeforeFirst('C');
                indcount = tstring2.Freq('.');
                for(int i=0; i<indcount;i++)convString<<"\t";
            tempString.Replace("Call ","%Call ");
            tempString = tempString.AfterFirst('%');
            convString << conv_p_callfn(tempString)<<"\n";
      }
      else if(tempString.Contains("Variable"))
    {
            tstring2 = tempString.BeforeFirst('V');
            indcount = tstring2.Freq('.');
            for(int i=0; i<indcount;i++)convString<<"\t";
tempString.Replace("Variable ","%Variable ");
            tempString = tempString.AfterFirst('%');
            convString << conv_variable(tempString)<<"\n";
      }
      else if(tempString.Contains("else if "))
    {
            tstring2 = tempString.BeforeFirst('e');
            indcount = tstring2.Freq('.');
            endindcount = indcount;
            for(int i=0; i<indcount;i++)convString<<"\t";
tempString.Replace("else if ","%else if ");
            tempString = tempString.AfterFirst('%');
            convString << conv_p_elseif(tempString);
            for(int i=0; i<indcount;i++)convString<<"\t";
            convString<<"{\n";
      }
      else if(tempString.Contains("if "))
    {
            tstring2 = tempString.BeforeFirst('i');
            indcount = tstring2.Freq('.');
            endindcount = indcount;
            for(int i=0; i<indcount;i++)convString<<"\t";
            tempString.Replace("if ","%if ");
            tempString = tempString.AfterFirst('%');
            convString << conv_p_if(tempString);
            for(int i=0; i<indcount;i++)convString<<"\t";
            convString<<"{\n";
      }
      else if(tempString.Contains("else "))
    {
            tstring2 = tempString.BeforeFirst('e');
            indcount = tstring2.Freq('.');
            endindcount = indcount;
            for(int i=0; i<indcount;i++)convString<<"\t";
            convString <<"else \n";
            for(int i=0; i<indcount;i++)convString<<"\t";
            convString<<"{\n";
      }
      else if(tempString.Contains("for loop from "))
    {
            tstring2 = tempString.BeforeFirst('f');
            indcount = tstring2.Freq('.');
            endindcount = indcount;
            for(int i=0; i<indcount;i++)convString<<"\t";
            tempString.Replace("for loop from ","%for loop from ");
            tempString = tempString.AfterFirst('%');
            convString << conv_p_forloop(tempString);
            for(int i=0; i<indcount;i++)convString<<"\t";
            convString<<"{\n";
      }
      else if(tempString.Contains("while loop until "))
    {
```

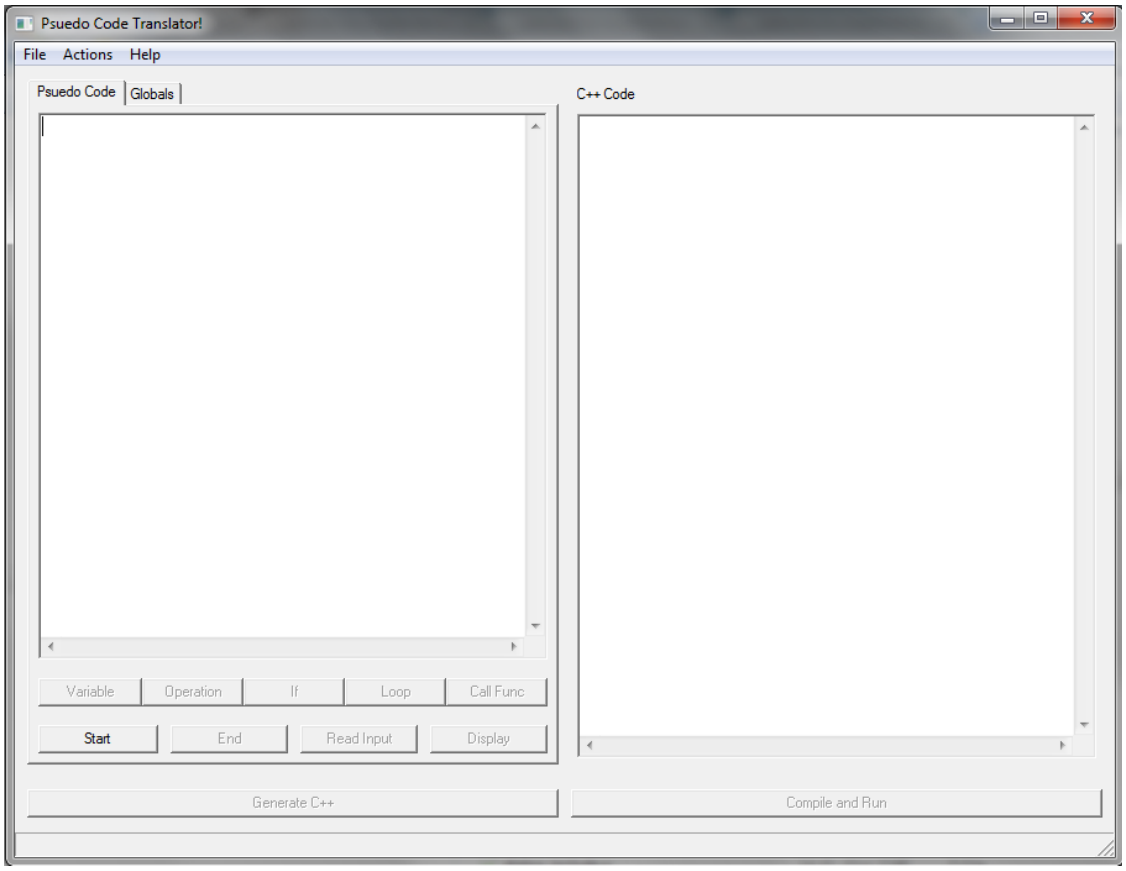
```
                    tstring2 = tempString.BeforeFirst('w');
                    indcount = tstring2.Freq('.');
                    endindcount = indcount;
                    for(int i=0; i<indcount;i++)convString<<"\t";
                tempString.Replace("while loop until ","%while loop until ");
                    tempString = tempString.AfterFirst('%');
                    convString << conv_p_whileloop(tempString);
                    for(int i=0; i<indcount;i++)convString<<"\t";
                    convString<<"{\n";
              }
              else if(tempString.Contains("Function "))
              {
                    endindcount=0;
                    convString <<  conv_p_func(tempString);
              }
              else if(tempString.Matches("*End *"))
              {
                    for(int i=0; i<endindcount;i++)convString<<"\t";
                    convString << "}\n";
                    endindcount--;
              }
              else if(tempString.Matches("*?. *"))
              {
                    tstring2 = tempString.BeforeFirst(' ');
                    indcount = tstring2.Freq('.');
                    for(int i=0; i<indcount;i++)convString<<"\t";
                    tempString = tempString.AfterFirst(' ');
                    convString << tempString <<";\n";
              }
           pString= pString.AfterFirst('$');
        }

        return convString;
}

wxString conv_variable(wxString input) //converts input of form "Variable
vame[23] as datatype*"  to datatype* vname[23];
{
      wxString output,vname,datatype;
      input = input.AfterFirst(' '); //input contains "vame[23] as
datatype*"
      vname = input.BeforeFirst(' ');
      input = input.AfterFirst(' ');//input contains "as datatype*
      datatype= input.AfterFirst(' ');
      output.Empty();
      output<<datatype<<" "<<vname<<";";
      return output;
}

wxString conv_g_func(wxString input)
{
      wxString output,fname,rtype,params;
      input = input.AfterFirst(' ');
      fname = input.BeforeFirst(' ');
      input = input.AfterFirst(' '
      input = input.AfterFirst(' ');
      input.Replace("returns","%returns");
      params = input.BeforeFirst('%');
      if(params.Contains(wxT("none"))) params.Empty();
      input = input.AfterFirst('%');
      rtype = input.AfterFirst(' ');
```

```
		output.Empty();
		output<< rtype <<" " << fname << "(" << params <<");";
	}

	wxString conv_class(wxString input)
	{
		wxString output,classname,temporary;
		output.Empty();
		classname = input.BeforeFirst('$');
		classname = classname.BeforeFirst(':');
		input = input.AfterFirst('$');
		temporary = input.BeforeFirst('$');
		input= input.AfterFirst('$');
		while(input.Contains(wxT("public:")))
		{
			temporary = input.BeforeFirst('$');
			if(temporary.Contains(wxT("public:"))) break;
			else if(temporary.Contains(wxT("Variable")))
			{
				temporary.Replace("\t ","");
				output<<"\t "<< conv_variable(temporary)<<"\n";
			}
			else if(temporary.Contains(wxT("Function")))
			{
				temporary.Replace("\t ","");
				output<<"\t " << conv_g_func(temporary)<<"\n";
			}
			input = input.AfterFirst('$');
		}
		temporary = input.BeforeFirst('$');
		input = input.AfterFirst('$');
		while(input.Contains(wxT("End")))
		{
			temporary = input.BeforeFirst('$');
			if(temporary.Contains(wxT("End"))) break;
			else if(temporary.Contains(wxT("Variable")))
			{
				temporary.Replace("\t ","");
				output<<"\t "<< conv_variable(temporary)<<"\n";
			}
			else if(temporary.Contains(wxT("Function")))
			{
				temporary.Replace("\t ","");
				output<<"\t " << conv_g_func(temporary)<<"\n";
			}
			input = input.AfterFirst('$');
		}
		output<<"\n};";
		return output;
	}

	wxString conv_struct(wxString input)//input contains Structure
	structname:$\nvariable ...$\n
	{
		wxString output,structname,temporary;
		output.Empty();
		structname = input.BeforeFirst('$');//contains Structure structname:
		structname = structname.AfterFirst(' ');
		structname = structname.BeforeFirst(':');
		input = input.AfterFirst('$');
		output << "struct " <<structname;
```

```
      output<<"\n{\n";
      while(input.Contains(wxT("End")))
      {
            temporary = input.BeforeFirst('$');
            if(temporary.Contains(wxT("End"))) break;
            else if(temporary.Contains(wxT("Variable")))
            {
                  temporary.Replace("\t ","");
                  output<<"\t "<< conv_variable(temporary)<<"\n";
            }
            else if(temporary.Contains(wxT("Function")))
            {
                  temporary.Replace("\t ","");
                  output<<"\t " << conv_g_func(temporary)<<"\n";
            }
            input = input.AfterFirst('$');
      }//input contains End Struct $\n
      output<<"\n};";
      return output;
}

wxString conv_globals()//Parse Globals Tab Code
{
      wxString tempString,convString,gString;
      tempString.Empty();
      tempString=get_data_in_file();
      convString.Empty();
      tempString.Replace("@Begin\n","");
      gString= tempString.AfterFirst ('$');
      gString=gString.BeforeFirst('#');
      //the main parser:
      gString.Replace("\n","$\n");
      tempString.Empty();
      while(gString.Contains(wxT("$")))
      {
            tempString.Empty();
          tempString = gString.BeforeFirst('$');//tempstring stores a line
            //converting variable
            if(tempString.Contains(wxT("Variable")))
            convString << conv_variable(tempString)<<"\n\n";
            //Converting Class;
            else if (tempString.Contains(wxT("Class")))
            {
                  tempString = gString;
                  tempString.Replace("End Class","End Class%");
                  gString = tempString.AfterFirst('%');
                  tempString = tempString.BeforeFirst('%');
                  convString << conv_class(tempString)<<"\n\n";
         }
         //Converting Structure
         else if (tempString.Contains(wxT("Structure")))
         {
                  tempString = gString;
                  tempString.Replace("End Structure","End
Structure%");//Temp String will have everything
                  gString = tempString.AfterFirst('%');
                  tempString = tempString.BeforeFirst('%');
                  convString << conv_struct(tempString)<<"\n\n";
         }
         gString= gString.AfterFirst('$');
      }
```

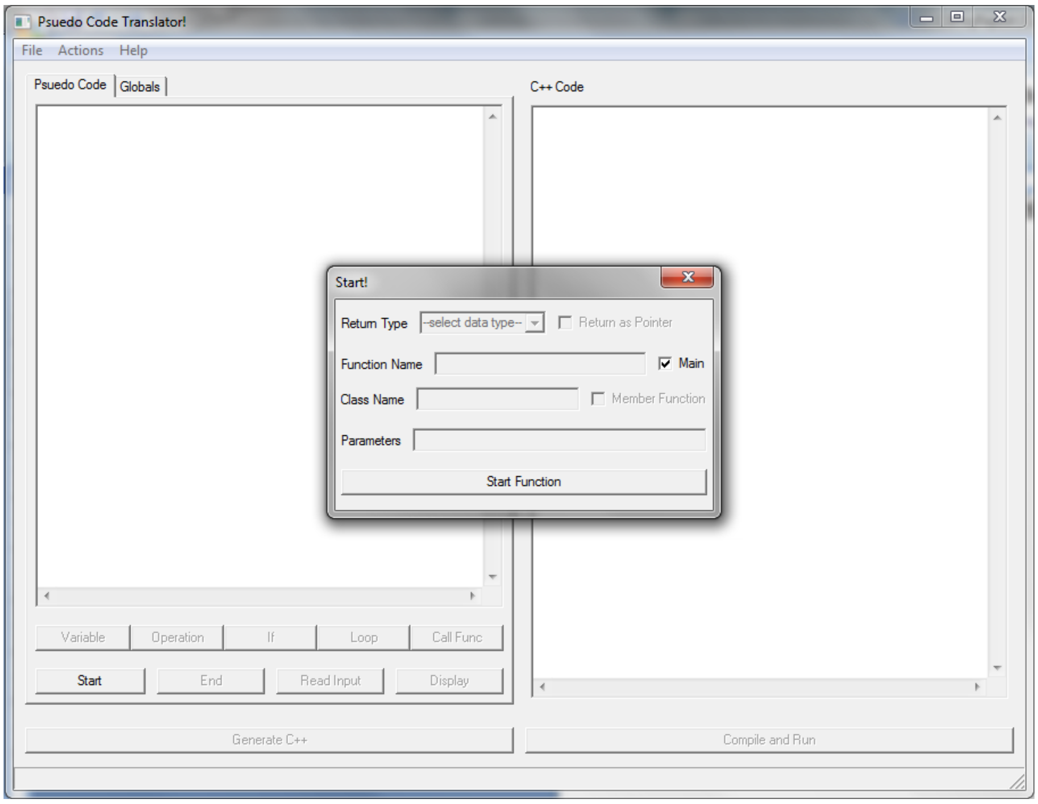
```
        return convString;
    }
```

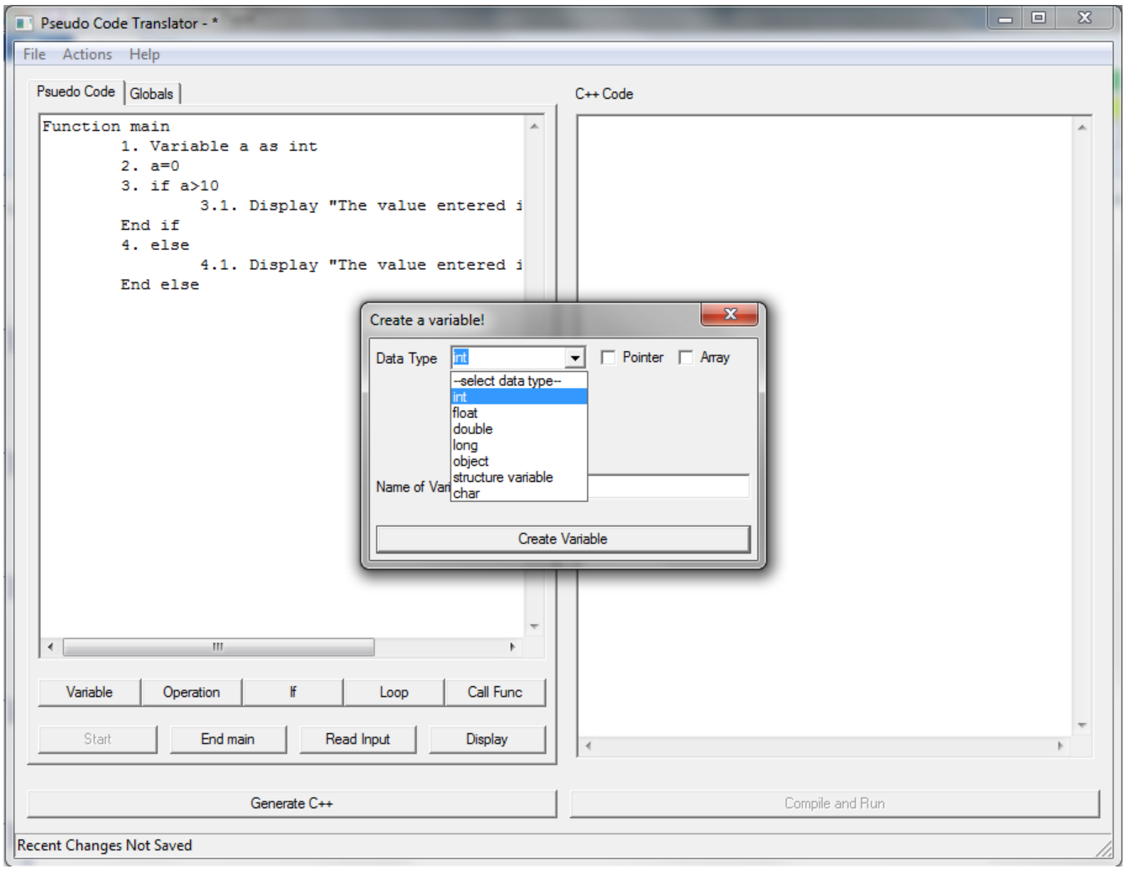
Screen Shots



The Main User Interface (MainFrame)

The Start Dialog

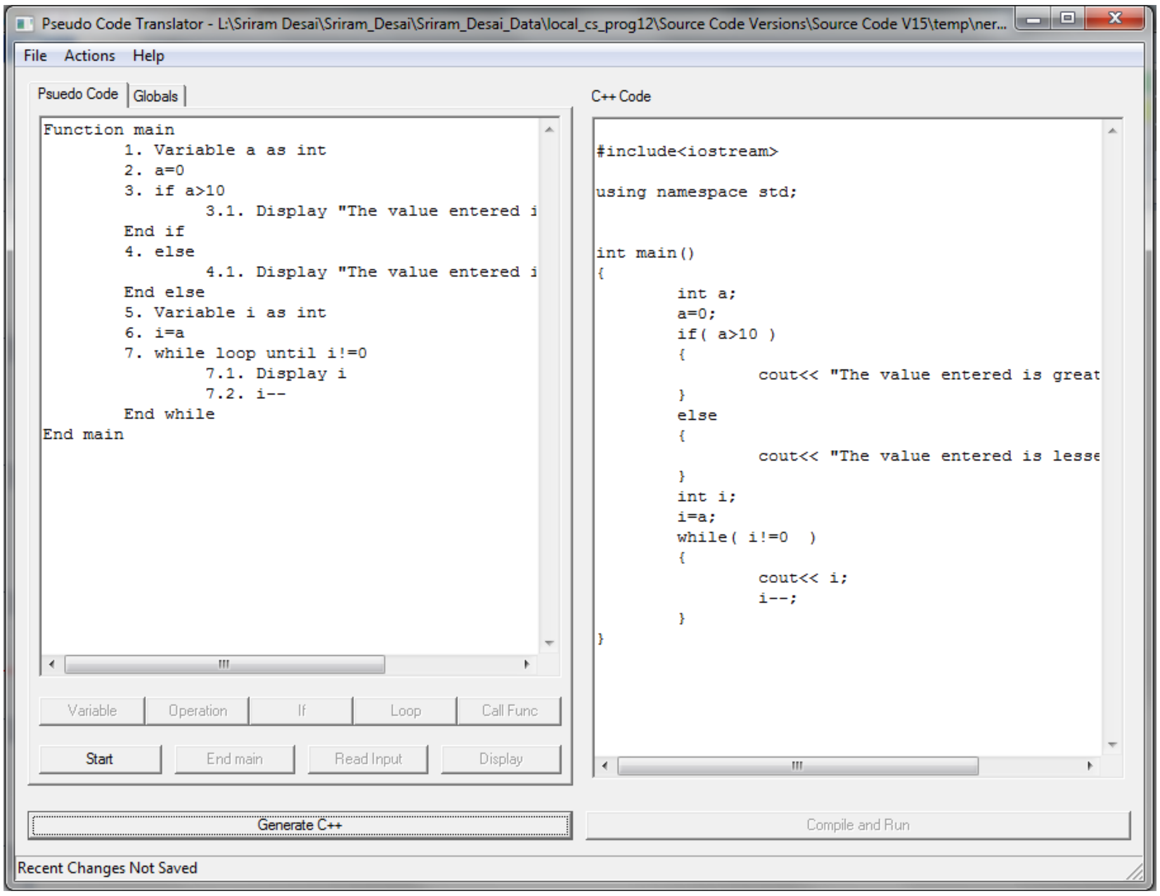




```
Function main
        1. Variable a as int
        2. a=0
        3. if a>10
                3.1. Display "The value entered i
        End if
        4. else
                4.1. Display "The value entered i
        End else
```

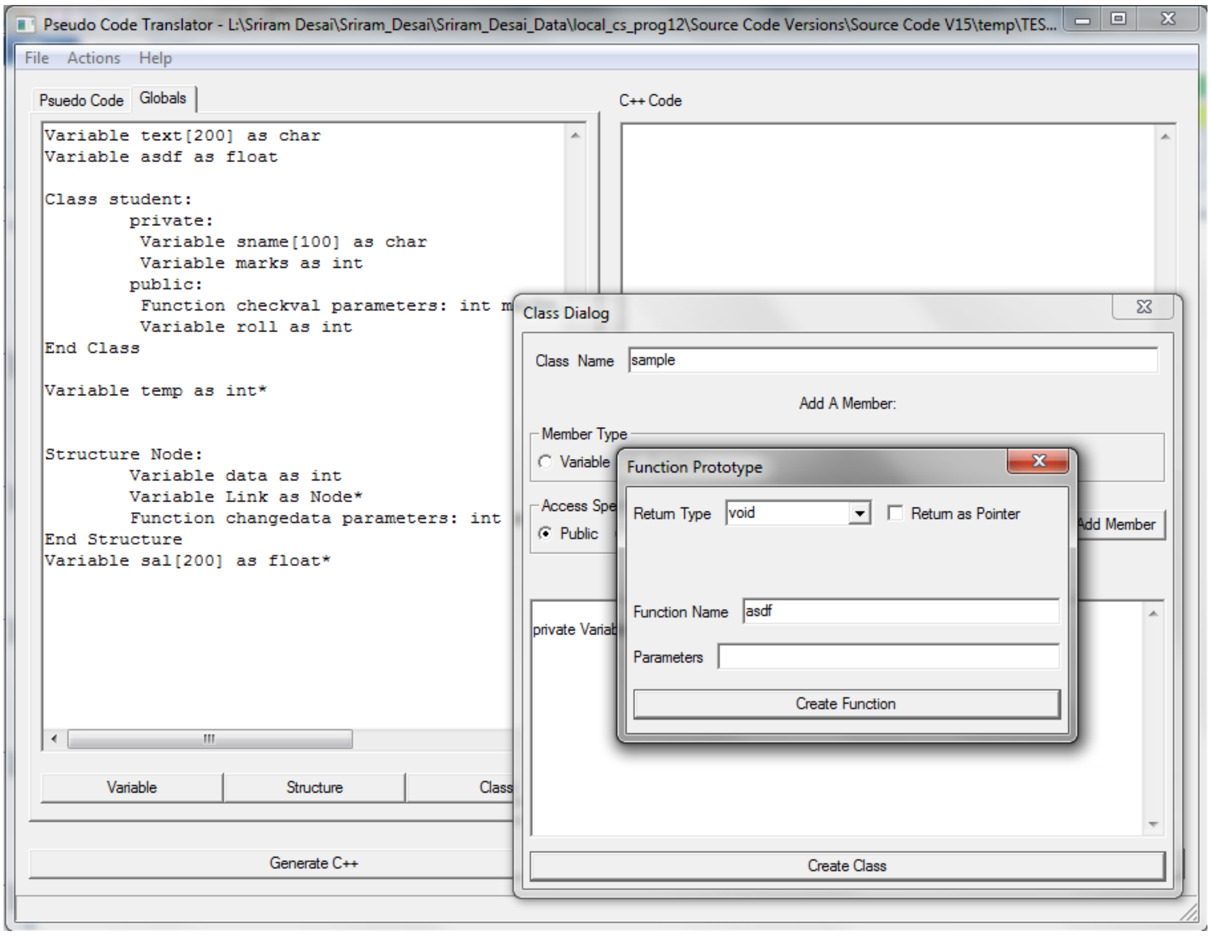
Creation of a Variable.

A completed program



```
Function main
        1. Variable a as int
        2. a=0
        3. if a>10
                3.1. Display "The value entered i
        End if
        4. else
                4.1. Display "The value entered i
        End else
        5. Variable i as int
        6. i=a
        7. while loop until i!=0
                7.1. Display i
                7.2. i--
        End while
End main
```

```
#include<iostream>

using namespace std;


int main()
{
        int a;
        a=0;
        if( a>10 )
        {
                cout<< "The value entered is great
        }
        else
        {
                cout<< "The value entered is lesse
        }
        int i;
        i=a;
        while( i!=0  )
        {
                cout<< i;
                i--;
        }
}
```



```
Variable text[200] as char
Variable asdf as float

Class student:
        private:
         Variable sname[100] as char
         Variable marks as int
        public:
         Function checkval parameters: int m
         Variable roll as int
End Class

Variable temp as int*


Structure Node:
        Variable data as int
        Variable Link as Node*
        Function changedata parameters: int
End Structure
Variable sal[200] as float*
```

Class Name: sample

Return Type: void
[ ] Return as Pointer
Function Name: asdf
Parameters:

Creation of a class

Scope for Improvement

The project is still far from perfection. Though we have been able to implement a lot of features, there is still a large scope for improvement. The areas that still require improvement are as follows:

- The editing system: Currently we do not support editing parts of the pseudo code through the GUI, all editing must be done by using the textbox alone. We do not support editing of elements using the built in dialogs that are used to insert the elements into the pseudo code.
- Deletion: Currently we do not support deletion of statements perfectly. Currently, once a statement is deleted, the indentation as well as step numbering of the remaining statements remains the same, and do not change in response to the deletion.
- Exporting pseudo code: The program at present offers no method of exporting the pseudo code, though users can save their work and resume later, the saved file is specifically structured for opening only with our program, hence to get only the pseudo code part, one has to manually copy it from the text areas in the program. We do however support exporting the converted C++ code.
- Compile and Run: In the application, we have made provisions for compiling and running the converted C++ code, but due to the difficulties in ensuring a compiler is already installed on the computer, we have removed that part of the code and disabled those buttons in the current version of the project, but we will be addressing this issue in future versions of the program.
- GUI: Though the app currently uses a modern UI system, we felt that we could improve the UI and add features such as Drag and Drop, instead of the current quick insert buttons.
- In app- help system: The program currently does not have a help system in built with the program, though we do have documentation for the program and its usage. We will be incorporating this in future versions.

We are keen on continuing development of this project, as it proves useful in helping new programmers pick up concepts easily. We will work to add in the above mentioned improvements to enhance the functionality even further.

References

- **wxWidgets 2.4.2 Documentation**
- http://wxwidgets.org/
- http://forums.wxwidgets.org/
- http://devpaks.org/ - DevPaks repository for Dev C++ packages
- http://stackoverflow.com/questions/tagged/wxwidgets
- C++: The Complete Reference - Tata McGraw-Hill Book
- Computer Science with C++ - Class 12 text book
- http://www.codeproject.com/Articles/11515/Introduction-to-wxWidgets
- http://sourceforge.net/apps/mediawiki/wxformbuilder/