Proyecto Integrador - 2do Semestre

Generated by Doxygen 1.8.4

Tue Jun 4 2013 00:34:21

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

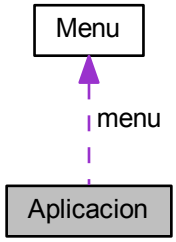
Chapter 4

Class Documentation

4.1 Aplicacion Class Reference

`#include <Aplicacion.h>`

Collaboration diagram for Aplicacion:



Public Member Functions

- **Aplicacion** ()
- ∼**Aplicacion** ()
- void **iniciar** ()
- void **addPregunta** (**Pregunta** ∗input)

Private Member Functions

- void **Encuesta** ()
- void **Registros** ()

Private Attributes

- bool **final**
- vector< **Pregunta** * > **preguntas**
- **Menu menu**

4.1.1 Detailed Description

Definition at line 13 of file Aplicacion.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Aplicacion::Aplicacion ( )

Definition at line 18 of file Aplicacion.cpp.

```
19 {
20     menu.addOpcion("Encuesta");
21     menu.addOpcion("Consulta");
22     final =true;
23 }
```

4.1.2.2 Aplicacion::~Aplicacion ( )

Definition at line 13 of file Aplicacion.cpp.

```
14 {
15
16 }
```

4.1.3 Member Function Documentation

4.1.3.1 void Aplicacion::addPregunta (**Pregunta** * *input*)

Definition at line 58 of file Aplicacion.cpp.

```
59 {
60     preguntas.push_back(input);
61 }
```

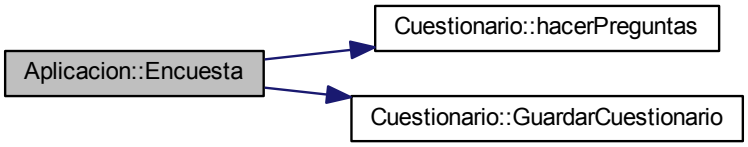
Here is the caller graph for this function:

4.1.3.2 void Aplicacion::Encuesta ( ) [private]

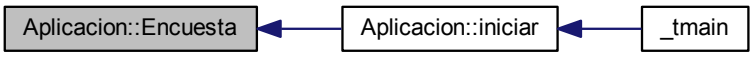
Definition at line 44 of file Aplicacion.cpp.

```
45 {
46     Cuestionario cuestionario(preguntas);//se crea un cuestionario donde se aran las preguntas y se
       responderan, tambicon este objeto se guardaran
47     //for (size_t i=0;i<preguntas.size();i++) {cuestionario.addPregunta(preguntas[i]);} //En este ciclo se
       an todas las preguntas del vector al cuestionario
48
49     cuestionario.hacerPreguntas();//con este metodo se imprimien y se contestan las preguntas del
       cuestionario
50     if (cuestionario.GuardarCuestionario()) //Aqui intenta guardar el cuestionario en un archivo de texto
       "respuestas.txt"
51     {
52         cout<<"Encuesta guardada. Gracias :)"; //si se logra guardar, entonces se imprime este mensaje
53         final=false; //se asigna esta variable global a 0, para que el hilo principal salga del ciclo y
       entonces pueda cerrarse el programa
54     }
55     else cout<<"Error al guardar la encuesta, Contacte al Encargado"; //Se imprime este mensaje y no se
       cierra el programa, cuando hay un error al guardar la encuesta
56 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



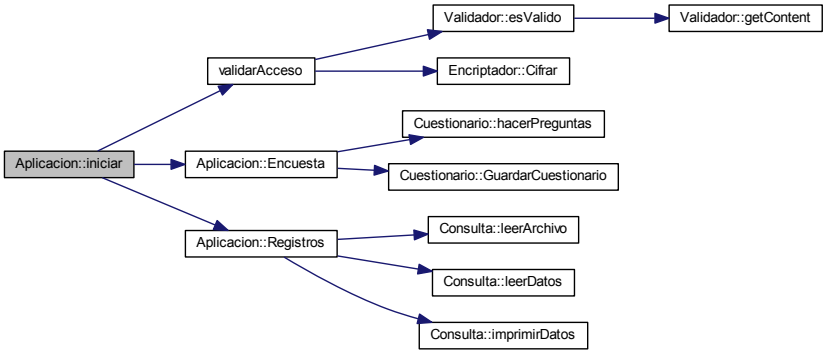
4.1.3.3 void Aplicacion::iniciar ( )

Definition at line 63 of file Aplicacion.cpp.

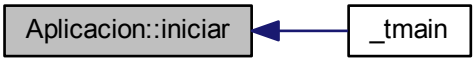
```
64 {
65     if(!validarAcceso())return ; //Aqui se checa que la llave este puesta, si no esta se sale del programa,
       si estontinua la ejecucincontrada, Favor de elegir la opcion adecuada\n ";
    thread * hiloElegido = nullptr;
    switch (menu.SeleccionarOpcion())
    {
    case 1:
```

```
        hiloElegido = new thread(&Aplicacion::Encuesta,this);
        break;
    case 2:
        hiloElegido = new thread(&Aplicacion::Registros,this);
        break;
    default:
        final=false;
        break;
    }


    //thread hiloEjecucion(&Aplicacion::Registros,this); //Aqui se llama asronamente un hilo de
       ejecucillamando a la funcion previamente definida llamada "AplicacionDeEncuesta"
    this_thread::sleep_for(std::chrono::milliseconds(3000)); //Se espera 3 segundos antes de entrar al
       ciclo infinito
    while (validarAcceso()&&final) //Este es el ciclo While que se realiza infinitamente, hasta que la
       encuesta se termine (pues la variable final queda en 0 y dado que 1^0=0)
    {
        this_thread::sleep_for(std::chrono::milliseconds(3000));//El proceso duerme por 3 segundos antes de
       volver a validar
    }
    if (hiloElegido!=nullptr)
    {
        hiloElegido->detach(); //Se destruye el hilo de ejecucie AplicacionDeEncuesta despues de salir del
       ciclo
    }

    /*string lala[10];
    lala->size();*/
    //cin.getline(new char,1);
}

66      cout<<"Clave encontrada, Favor de elegir la opcion adecuada\n ";
67      thread * hiloElegido = nullptr;
68      switch (menu.SeleccionarOpcion())
69      {
70      case 1:
71          hiloElegido = new thread(&Aplicacion::Encuesta,this);
72          break;
73      case 2:
74          hiloElegido = new thread(&Aplicacion::Registros,this);
75          break;
76      default:
77          final=false;
78          break;
79      }
80
81
82      //thread hiloEjecucion(&Aplicacion::Registros,this); //Aqui se llama asronamente un hilo de
        ejecucillamando a la funcion previamente definida llamada "AplicacionDeEncuesta"
83      this_thread::sleep_for(std::chrono::milliseconds(3000)); //Se espera 3 segundos antes de entrar al
        ciclo infinito
84      while (validarAcceso()&&final) //Este es el ciclo While que se realiza infinitamente, hasta que la
        encuesta se termine (pues la variable final queda en 0 y dado que 1^0=0)
85      {
86          this_thread::sleep_for(std::chrono::milliseconds(3000));//El proceso duerme por 3 segundos antes de
        volver a validar
87      }
88      if (hiloElegido!=nullptr)
89      {
90          hiloElegido->detach(); //Se destruye el hilo de ejecucie AplicacionDeEncuesta despues de salir del
        ciclo
91      }
92
93      /*string lala[10];
94      lala->size();*/
95      //cin.getline(new char,1);
96  }
```

Here is the call graph for this function:



Here is the caller graph for this function:

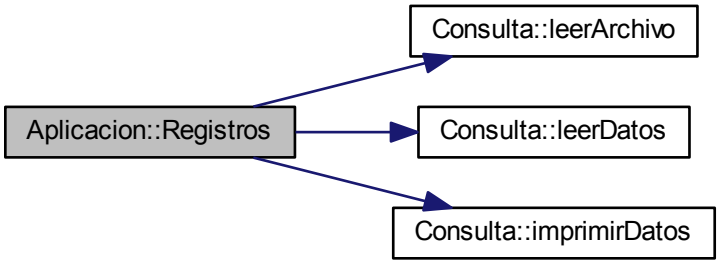


4.1.3.4 void Aplicacion::Registros ( ) [private]

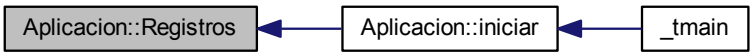
Definition at line 25 of file Aplicacion.cpp.

```
26 {
27     Aplicacion();
28     Consulta consulta(preguntas);
29     if (consulta.leerArchivo("respuesta.txt"))
30     {
31         consulta.leerDatos();
32         consulta.imprimirDatos();
33         cin.getline(new char,1);
34         cin.ignore();
35         final=false;
36 
37     }
38     else
39     {
40         cout<<"Error al leer el archivo de las respuestas"<<endl;
41     }
42 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 bool Aplicacion::final [private]

Definition at line 21 of file Aplicacion.h.

4.1.4.2 Menu Aplicacion::menu [private]

Definition at line 23 of file Aplicacion.h.

4.1.4.3 vector<Pregunta*> Aplicacion::preguntas [private]

Definition at line 22 of file Aplicacion.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Aplicacion.h**
- ProyectoIntegrador2/**Aplicacion.cpp**

4.2 Consulta Class Reference

`#include <Consulta.h>`

Public Member Functions

- **Consulta** ()
- **Consulta** (vector< **Pregunta** * > **preguntas**)
- ~**Consulta** ()
- void **addPregunta** (**Pregunta** *input)
- bool **leerArchivo** (string filename)
- void **leerDatos** ()
- void **imprimirDatos** ()

Private Attributes

- ifstream **archivo**
- vector< vector< string > > **Registros**
- vector< **Pregunta** * > **preguntas**

4.2.1 Detailed Description

Definition at line 14 of file Consulta.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Consulta::Consulta ( )

Definition at line 69 of file Consulta.cpp.

```
70 {
71
72 }
```

4.2.2.2 Consulta::Consulta (vector< **Pregunta** * > *preguntas*)

Definition at line 74 of file Consulta.cpp.

```
75 {
76     this->preguntas=preguntas;
77 }
```

4.2.2.3 Consulta::~Consulta ( )

Definition at line 64 of file Consulta.cpp.

```
65 {
66
67 }
```

4.2.3 Member Function Documentation

4.2.3.1 void Consulta::addPregunta (Pregunta * *input*)

Definition at line 9 of file Consulta.cpp.

```
10 {
11     preguntas.push_back(input);
12 }
```

4.2.3.2 void Consulta::imprimirDatos ( )

Definition at line 48 of file Consulta.cpp.

```
49 {
50     for (int i = 0; i < Registros.size(); i++)
51     {
52         cout << "\nCuestionario #"<<i+1<<endl;
53         cout<<"Fecha: "<<Registros[i][0]<<endl;
54         //cout<<preguntas.size()<<endl;
55         for (int j = 0; j < preguntas.size(); j++)
56         {
57             preguntas[j]->imprimirPregunta();
58             cout << Registros[i][j+1] << endl;
59         }
60
61     }
62 }
```

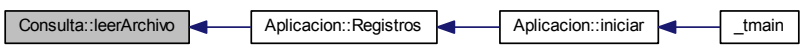
Here is the caller graph for this function:

4.2.3.3 bool Consulta::leerArchivo (string *filename*)

Definition at line 14 of file Consulta.cpp.

```
15 {
16     archivo.open(filename,ios::binary|ios::in);
17     //fs.read(buffer,1000000);
18     if(!archivo){
19         return false;
20     }
21     return true;
22 }
```

Here is the caller graph for this function:

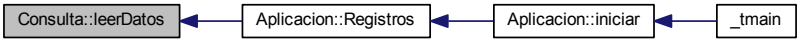


4.2.3.4 void Consulta::leerDatos ( )

Definition at line 24 of file Consulta.cpp.

```
25 {
26     vector<string> Respuestas;
27     string temporal="";
28     char buffer;
29     while (archivo.good())
30     {
31         buffer=archivo.get();
32         if(buffer=='\t'){
33             Respuestas.push_back(temporal);
34             //cout<<temporal<<endl;
35             temporal="";
36             continue;
37         }
38         if(buffer=='\n'){
39             Registros.push_back(Respuestas);
40             Respuestas.clear();
41             //cout<<"Nuevo Registro"<<endl;
42             continue;
43         }
44         temporal+=buffer;
45     }
46 }
```

Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 ifstream Consulta::archivo [private]

Definition at line 25 of file Consulta.h.

4.2.4.2 vector<Pregunta*> Consulta::preguntas [private]

Definition at line 27 of file Consulta.h.

4.2.4.3 vector<vector<string> > Consulta::Registros [private]

Definition at line 26 of file Consulta.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Consulta.h**
- ProyectoIntegrador2/**Consulta.cpp**

4.3 Cuestionario Class Reference

`#include <Cuestionario.h>`

Public Member Functions

- **Cuestionario** ()
- **Cuestionario** (vector< **Pregunta** * > **preguntas**)
- ~**Cuestionario** ()
- bool **GuardarCuestionario** ()
- void **hacerPreguntas** ()
- void **addPregunta** (**Pregunta** *)

Private Attributes

- std::vector< **Pregunta** * > **preguntas**

4.3.1 Detailed Description

Definition at line 6 of file Cuestionario.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Cuestionario::Cuestionario ( )

Definition at line 14 of file Cuestionario.cpp.

```
15 {
16
17 }
```

4.3.2.2 Cuestionario::Cuestionario (vector< Pregunta * > *preguntas*)

Definition at line 19 of file Cuestionario.cpp.

```
20 {
21     this->preguntas=preguntas;
22 }
```

4.3.2.3 Cuestionario::~Cuestionario ( )

Definition at line 24 of file Cuestionario.cpp.

```
25 {
26
27 }
```

4.3.3 Member Function Documentation

4.3.3.1 void Cuestionario::addPregunta (Pregunta * *input*)

Definition at line 71 of file Cuestionario.cpp.

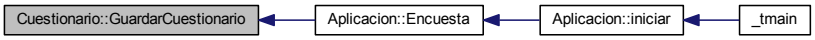
```
71                                          {
72     preguntas.push_back(input);
73
74 }
```

4.3.3.2 bool Cuestionario::GuardarCuestionario ( )

Definition at line 29 of file Cuestionario.cpp.

```
30 {
31     typedef std::chrono::system_clock Clock;
32     auto now = Clock::now();
33     std::time_t now_c = Clock::to_time_t(now);
34     struct tm *parts = std::localtime(&now_c);
35     int anio= 1900 + parts->tm_year;
36     int month= 1    + parts->tm_mon;
37     int day = parts->tm_mday ;
38 
39     ofstream myfile;
40     myfile.open("respuesta.txt",ios::out | ios::app);
41     string respues = "";
42     if (myfile.is_open())
43     {
44         for (size_t i=0;i<preguntas.size();i++)
45         {
46             respues+=preguntas[i]->getRespuesta()+"\t";
47         }
48         myfile << 1900 + parts->tm_year  << "_";
49         myfile << 1    + parts->tm_mon   << "_";
50         myfile <<        parts->tm_mday  << "\t";
51         myfile << respues <<"\n";
52         myfile.close();
53         return true;
54     }
55     else cout << "No se puede guardar";
56     return false;
57 }
```

Here is the caller graph for this function:

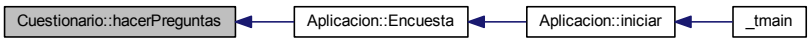


4.3.3.3 void Cuestionario::hacerPreguntas ( )

Definition at line 59 of file Cuestionario.cpp.

```
60 {
61 
62     int numeroPreguntas = this->preguntas.size();
63 //  cout<<preguntas.size()<<"eltama~no\n";
64     for (int contadorPreguntas=0;contadorPreguntas<numeroPreguntas;contadorPreguntas++)
65     {
66         preguntas[contadorPreguntas]->imprimirPregunta();
67         preguntas[contadorPreguntas]->responder();
68     }
69 }
```

Here is the caller graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 std::vector<Pregunta*> Cuestionario::preguntas [private]

Definition at line 16 of file Cuestionario.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Cuestionario.h**
- ProyectoIntegrador2/**Cuestionario.cpp**

4.4 Encriptador Class Reference

`#include <Encriptador.h>`

Public Member Functions

- **Encriptador** (std::string claveE, std::string claveD)
- ~**Encriptador** ()
- std::string **Cifrar** (std::string input)
- std::string **Decifrar** (std::string input)

Private Attributes

- std::string **claveEncriptacion**
- std::string **claveDesencriptacion**
- int **tamanoClave**

4.4.1 Detailed Description

Definition at line 16 of file Encriptador.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Encriptador::Encriptador (std::string *claveE*, std::string *claveD*)

Definition at line 51 of file Encriptador.cpp.

```
52 {
53     if (claveE.length() != claveD.length())
54     {
55         return;
56     }
57     claveEncriptacion=claveE;
58     claveDesencriptacion=claveD;
59     tamanoClave = claveD.length()-1;
60 }
```

4.4.2.2 Encriptador::~Encriptador ( )

Definition at line 46 of file Encriptador.cpp.

```
47 {
48
49 }
```

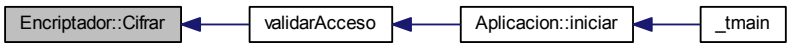
4.4.3 Member Function Documentation

4.4.3.1 std::string Encriptador::Cifrar (std::string *input*)

Definition at line 29 of file Encriptador.cpp.

```
30 {
31     for (size_t i =0;i<input.length();i++)
32     {
33         int posE = claveDesencriptacion.find(input[i]);
34         int posF = posE+i;
35         if (posF>tamanoClave) posF = ((posE+i)%tamanoClave)-1;
36         if(posE != std::string::npos)input[i]=claveEncriptacion[posF];
37         else input[i]='#';
38         //std::cout<<posF<<"<Pf Tc>"<<tamanoClave<<std::endl;
39     }
40
41
42
43     return input;
44 }
```

Here is the caller graph for this function:



4.4.3.2 std::string Encriptador::Decifrar (std::string *input*)

Definition at line 10 of file Encriptador.cpp.

```
11 {
12     for (size_t i =0;i<input.length();i++)
13     {
14         int posE = claveEncriptacion.find(input[i]);
15         int posF = posE - i;
16         if (posF<0)
17         {
18             posF %= tamanoClave;
19             posF += tamanoClave;
20             posF++;
21         }
22         if(posE != std::string::npos)input[i]=claveDesencriptacion[posF];
23         else input[i]='#';
24         //std::cout<<posF<<"<Pf Tc>"<<tamanoClave<<std::endl;
25     }
26     return input;
27 }
```

4.4.4 Member Data Documentation

4.4.4.1 std::string Encriptador::claveDesencriptacion [private]

Definition at line 25 of file Encriptador.h.

4.4.4.2 std::string Encriptador::claveEncriptacion [private]

Definition at line 24 of file Encriptador.h.

4.4.4.3 int Encriptador::tamanoClave [private]

Definition at line 26 of file Encriptador.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Encriptador.h**
- ProyectoIntegrador2/**Encriptador.cpp**

4.5 Menu Class Reference

`#include <Menu.h>`

Public Member Functions

- **Menu** ()
- **Menu** (vector< string > **opciones**)
- int **SeleccionarOpcion** ()
- void **addOpcion** (string opcion)

Private Attributes

- vector< string > **opciones**
- int **opc**

4.5.1 Detailed Description

Definition at line 10 of file Menu.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Menu::Menu ( )

Definition at line 9 of file Menu.cpp.

```
10 {
11     opc=0;
12 }
```

4.5.2.2 Menu::Menu (vector< string > *opciones*)

Definition at line 14 of file Menu.cpp.

```
15 {
16     Menu();
17     this->opciones=opciones;
18 }
```

4.5.3 Member Function Documentation

4.5.3.1 void Menu::addOpcion (string *opcion*)

Definition at line 55 of file Menu.cpp.

```
56 {
57     opciones.push_back(opcion);
58 }
```

4.5.3.2 int Menu::SeleccionarOpcion ( )

Definition at line 20 of file Menu.cpp.

```
21 {
22     //int opc=0; //opcion del Menu
23     while (opc!=(opciones.size()+1))//Imprimir el Menu hasta que se seleccione salir(opcion 3)
24     {
25         do{
26             system("cls");
27             //cout << "----------------------- Menu -------------------------" << endl ;
28             cout << "-------Elija la opcion que desea realizar---------: " << endl << endl ;
29             for (size_t i = 0; i < opciones.size(); i++)
30             {
31                 cout<<(i+1)<<".- "<<opciones[i]<<"."<<endl;
32             }
33             cout << (opciones.size()+1) <<".- Salir." << endl ;
34             cout << "Seleccionado: ";
35             //cin.ignore();
36             opc = _getche()-48;
37             if( opc<1 || opc >(int)(opciones.size()+1))
38             {
39                 cout << "Ese numero no es una opcion, vuelve a ingresar tu opcion." <<endl;
40                 opc=0;
41
42                 //system("pause");
43             }
44             else
45             {
46                 //6cin.ignore();
47                 return opc;
48             }
49         }while(true); //condicional
50         //cout << "Switch para las opciones";
51     }
52     return -1;
53 }
```

4.5.4 Member Data Documentation

4.5.4.1 int Menu::opc [private]

Definition at line 22 of file Menu.h.

4.5.4.2 vector<string> Menu::opciones [private]

Definition at line 21 of file Menu.h.

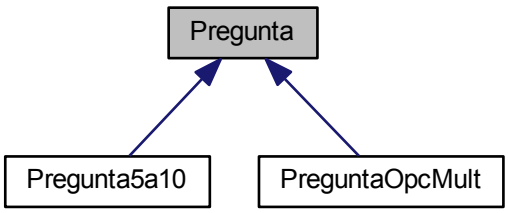
The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Menu.h**
- ProyectoIntegrador2/**Menu.cpp**

4.6 Pregunta Class Reference

`#include <Preguntas.h>`

Inheritance diagram for Pregunta:



Public Member Functions

- **Pregunta** (std::string)
- ~**Pregunta** ()
- virtual void **responder** ()
- virtual void **imprimirPregunta** ()
- std::string **getRespuesta** ()

Protected Attributes

- char **respuesta** [512]

Private Attributes

- std::string **pregunta**

4.6.1 Detailed Description

Definition at line 6 of file Preguntas.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Pregunta::Pregunta (std::string *pregunta*)

Definition at line 9 of file Preguntas.cpp.

```
10 {
11     this->pregunta=pregunta;
12     memset(respuesta,'\0',512);
13 }
```

4.6.2.2 Pregunta::~Pregunta ( ) [inline]

Definition at line 10 of file Preguntas.h.

```
10 {};
```

4.6.3 Member Function Documentation

4.6.3.1 std::string Pregunta::getRespuesta ( )

Definition at line 38 of file Preguntas.cpp.

```
39 {
40     return *(new string(respuesta));
41 }
```

4.6.3.2 void Pregunta::imprimirPregunta ( ) [virtual]

Reimplemented in **PreguntaOpcMult** (p. 27), and **Pregunta5a10** (p. 25).

Definition at line 33 of file Preguntas.cpp.

```
34 {
35     cout<<"\n>> "<<pregunta<<": ";
36 }
```

4.6.3.3 void Pregunta::responder ( ) [virtual]

Reimplemented in **PreguntaOpcMult** (p. 28), and **Pregunta5a10** (p. 25).

Definition at line 15 of file Preguntas.cpp.

```
16 {
17     //cin.ignore();
18 //  this->imprimirPregunta();
19     cin.getline(respuesta,512);
20     for (int i=0;i<512;i++)
21     {
22         if (respuesta[i]=='\0')
23         {
24             break;
25         }
26         if (respuesta[i]=='\t')
27         {
28             respuesta[i]=' ';
29         }
30     }
31 }
```

4.6.4 Member Data Documentation

4.6.4.1 std::string Pregunta::pregunta [private]

Definition at line 15 of file Preguntas.h.

4.6.4.2 char Pregunta::respuesta[512] [protected]

Definition at line 17 of file Preguntas.h.

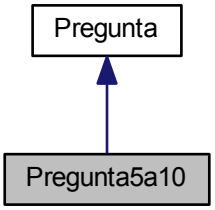
The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Preguntas.h**
- ProyectoIntegrador2/**Preguntas.cpp**

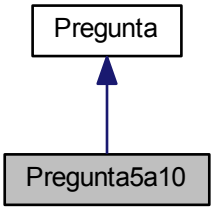
4.7 Pregunta5a10 Class Reference

`#include <Preguntas.h>`

Inheritance diagram for Pregunta5a10:



Collaboration diagram for Pregunta5a10:



Public Member Functions

- **Pregunta5a10** (std::string)
- ~**Pregunta5a10** ()
- void **responder** ()
- void **imprimirPregunta** ()

Additional Inherited Members

4.7.1 Detailed Description

Definition at line 20 of file Preguntas.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Pregunta5a10::Pregunta5a10 (std::string *input*)

Definition at line 43 of file Preguntas.cpp.

```
43                                          : Pregunta(input)
44 {
45
46 }
```

4.7.2.2 Pregunta5a10::~Pregunta5a10 ( ) [inline]

Definition at line 24 of file Preguntas.h.

```
24 {};
```

4.7.3 Member Function Documentation

4.7.3.1 void Pregunta5a10::imprimirPregunta ( ) [virtual]

Reimplemented from **Pregunta** (p. 23).

Definition at line 65 of file Preguntas.cpp.

```
66 {
67     //Pregunta::imprimirPregunta();
68     __super::imprimirPregunta();
69     cout<<" (5 a 10) | ";
70 }
```

4.7.3.2 void Pregunta5a10::responder ( ) [virtual]

Reimplemented from **Pregunta** (p. 23).

Definition at line 48 of file Preguntas.cpp.

```
49 {
50 //  imprimirPregunta();
51 __super::responder();
52 //cout<<respuesta<<"res\n";
53
54 istringstream ss(respuesta);
55 int x;
56 if (!(ss >> x) || (x>10 || x<5) || respuesta[2]!='\0' ) {
57
58     //cin.ignore();
59     cout<<"Numero Invalido\n";
60     //ss.ignore();
61     responder();
62 }
63 }
```

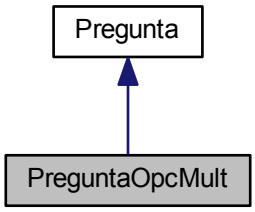
The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Preguntas.h**
- ProyectoIntegrador2/**Preguntas.cpp**

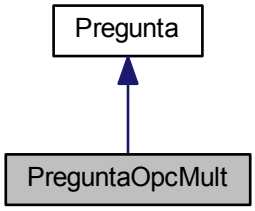
4.8 PreguntaOpcMult Class Reference

`#include <Preguntas.h>`

Inheritance diagram for PreguntaOpcMult:



Collaboration diagram for PreguntaOpcMult:



Public Member Functions

- **PreguntaOpcMult** (std::string, std::string)
- ~**PreguntaOpcMult** ()
- void **responder** ()
- void **imprimirPregunta** ()

Private Attributes

- int **numeroOpciones**
- std::vector< std::string > **opcions**

Additional Inherited Members

4.8.1 Detailed Description

Definition at line 30 of file Preguntas.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 PreguntaOpcMult::PreguntaOpcMult (std::string *input,* std::string *opci*)

Definition at line 72 of file Preguntas.cpp.

```
72                                                              : Pregunta(input)
73 {
74     //opciones=opci;
75     size_t noOpc;
76     numeroOpciones=0;
77     string container="";
78     for (noOpc=0;noOpc<opci.length();noOpc++)
79     {
80         if (opci[noOpc]=='\t' || noOpc==(opci.length()-1))
81         {
82             numeroOpciones++;
83             opcions.push_back(container);
84             container="";
85             continue;
86         }
87         container+=opci[noOpc];
88     }
89     //cout<<opci.length()<<"lasopcios\n";
90     //numeroOpciones=opci.length();
91 }
```

4.8.2.2 PreguntaOpcMult::~PreguntaOpcMult ( ) [inline]

Definition at line 34 of file Preguntas.h.

```
34 {};
```

4.8.3 Member Function Documentation

4.8.3.1 void PreguntaOpcMult::imprimirPregunta ( ) [virtual]

Reimplemented from **Pregunta** (p. 23).

Definition at line 114 of file Preguntas.cpp.

```
115 {
116     __super::imprimirPregunta();
117     cout<<endl;
118
119     for (int i=0;i<numeroOpciones;i++)
120     {
121         cout<<char(i+97)<<") "<<opcions[i]<<"\t";
122     }
123     cout<<"\n\n";
124 }
```

4.8.3.2 void PreguntaOpcMult::responder ( ) [virtual]

Reimplemented from **Pregunta** (p. 23).

Definition at line 93 of file Preguntas.cpp.

```
94 {
95     string opciones="";
96     for (int j = 0; j < numeroOpciones; j++)
97     {
98         opciones+=char(j+97);
99     }
100     char opc = _getch();
101     for (unsigned int i=0;i<opciones.length();i++)
102     {
103         if (opciones[i] == opc)
104         {
105             respuesta[0]=opc;
106             return;
107         }
108
109     }
110     responder();
111
112 }
```

4.8.4 Member Data Documentation

4.8.4.1 int PreguntaOpcMult::numeroOpciones [private]

Definition at line 39 of file Preguntas.h.

4.8.4.2 std::vector<std::string> PreguntaOpcMult::opcions [private]

Definition at line 40 of file Preguntas.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Preguntas.h**
- ProyectoIntegrador2/**Preguntas.cpp**

4.9 Validador Class Reference

`#include <Encriptador.h>`

Public Member Functions

- **Validador** (std::string filename)
- ~**Validador** ()
- bool **esValido** (std::string input)
- std::string **getContent** ()

Public Attributes

- std::string **path**
- int **size**

Private Attributes

- std::string **content**

4.9.1 Detailed Description

Definition at line 31 of file Encriptador.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Validador::Validador (std::string *filename*)

Definition at line 86 of file Encriptador.cpp.

```
87 {
88     for (char i='A';i<='Z';i++)
89     {
90         //std::cout<<i<<std::endl;
91         std::string tmpcontent;
92         path = "N:\\"+filename;
93         path[0]=i;
94         //std::cout<<"Checando en "+path<<std::endl;
95         std::ifstream archivoClave (path, std::ifstream::binary);
96         if (archivoClave.is_open())
97         {
98
99             content.assign( (std::istreambuf_iterator<char>(archivoClave) ),
100                  (std::istreambuf_iterator<char>()    ) );
101              break;
102              tmpcontent.assign(content.data());
103          }
104
105          //std::cout<<tmpcontent<<std::endl;
106
107     }
108 }
```

4.9.2.2 Validador::~Validador ( )

Definition at line 81 of file Encriptador.cpp.

```
82 {
83
84 }
```

4.9.3 Member Function Documentation

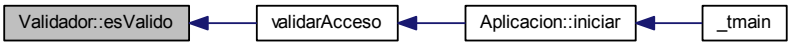
4.9.3.1 bool Validador::esValido (std::string *input*)

Definition at line 68 of file Encriptador.cpp.

```
69 {
70     std::string contenido = getContent();
71     if (input == contenido)
72     {
73         return true;
74     }
75     else
76     {
77         return false;
78     }
79 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



4.9.3.2 std::string Validador::getContent ( )

Definition at line 63 of file Encriptador.cpp.

```
64 {
65     return content;
66 }
```

Here is the caller graph for this function:

4.9.4 Member Data Documentation

4.9.4.1 std::string Validador::content [private]

Definition at line 44 of file Encriptador.h.

4.9.4.2 std::string Validador::path

Definition at line 37 of file Encriptador.h.

4.9.4.3 int Validador::size

Definition at line 38 of file Encriptador.h.

The documentation for this class was generated from the following files:

- ProyectoIntegrador2/**Encriptador.h**
- ProyectoIntegrador2/**Encriptador.cpp**

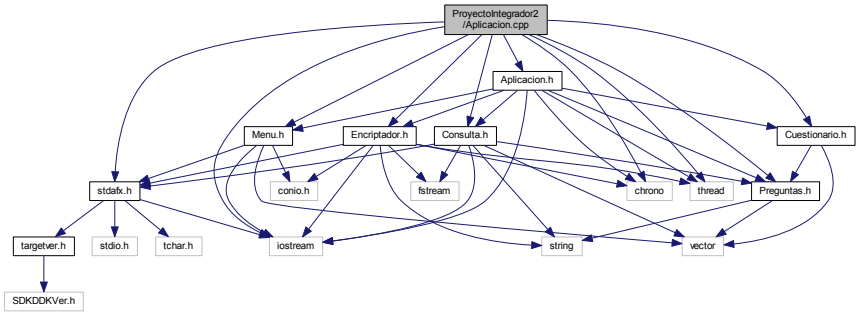
Chapter 5

File Documentation

5.1 ProyectoIntegrador2/Aplicacion.cpp File Reference

```
#include "stdafx.h"
#include <thread>
#include <iostream>
#include <chrono>
#include "Encriptador.h"
#include "Preguntas.h"
#include "Cuestionario.h"
#include "Consulta.h"
#include "Aplicacion.h"
#include "Menu.h"
```

Include dependency graph for Aplicacion.cpp:

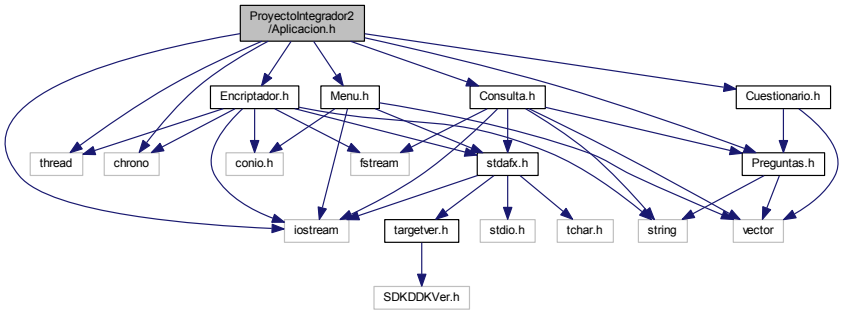


5.2 ProyectoIntegrador2/Aplicacion.h File Reference

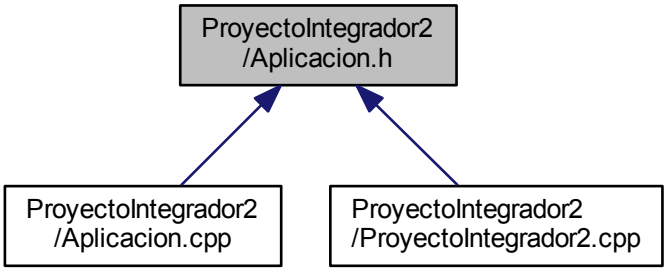
```
#include <thread>
```

```
#include <iostream>
#include <chrono>
#include "Encriptador.h"
#include "Preguntas.h"
#include "Cuestionario.h"
#include "Consulta.h"
#include "Menu.h"
```

Include dependency graph for Aplicacion.h:



This graph shows which files directly or indirectly include this file:



Classes

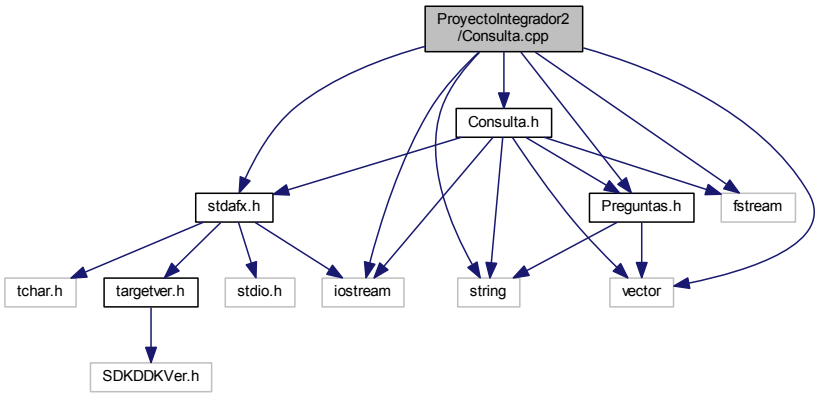
- class **Aplicacion**

5.3 ProyectoIntegrador2/Consulta.cpp File Reference

```
#include "stdafx.h"
```

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include "Preguntas.h"
#include "Consulta.h"
```

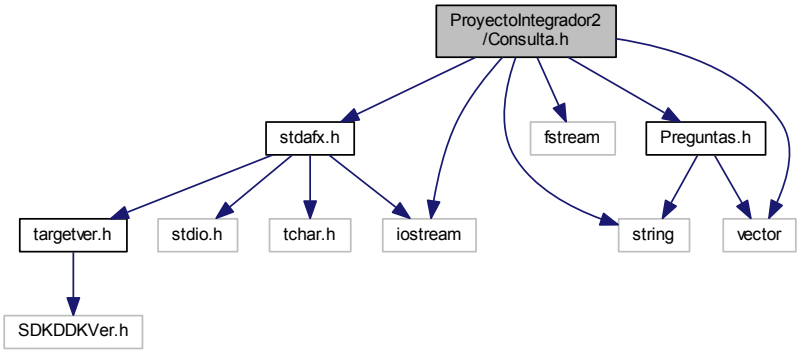
Include dependency graph for Consulta.cpp:



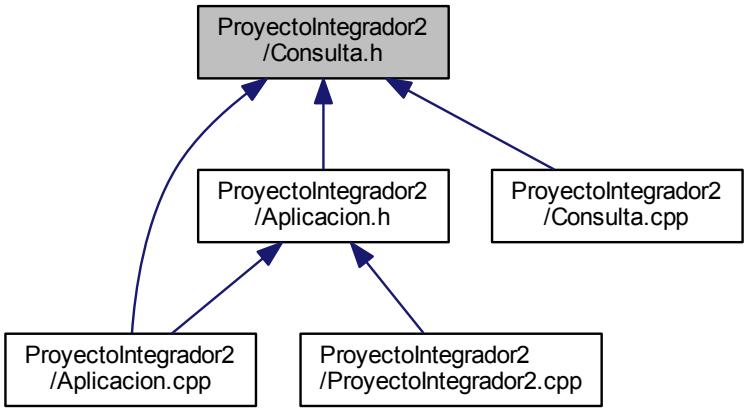
5.4 ProyectoIntegrador2/Consulta.h File Reference

```
#include "stdafx.h"
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include "Preguntas.h"
```

Include dependency graph for Consulta.h:



This graph shows which files directly or indirectly include this file:



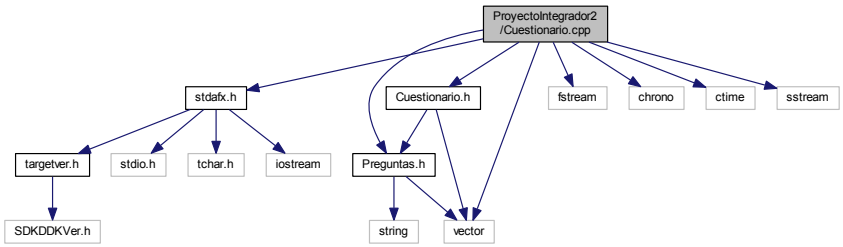
Classes

- class **Consulta**

5.5 ProyectoIntegrador2/Cuestionario.cpp File Reference

```
#include "stdafx.h"
#include "Cuestionario.h"
#include "Preguntas.h"
#include <vector>
#include <fstream>
#include <chrono>
#include <ctime>
#include <sstream>
```

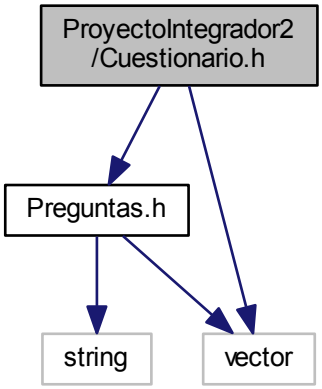
Include dependency graph for Cuestionario.cpp:



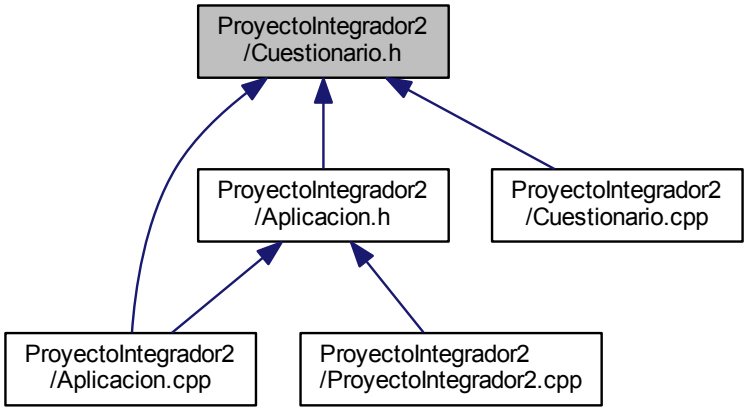
5.6 ProyectoIntegrador2/Cuestionario.h File Reference

```
#include "Preguntas.h"
#include <vector>
```

Include dependency graph for Cuestionario.h:



This graph shows which files directly or indirectly include this file:



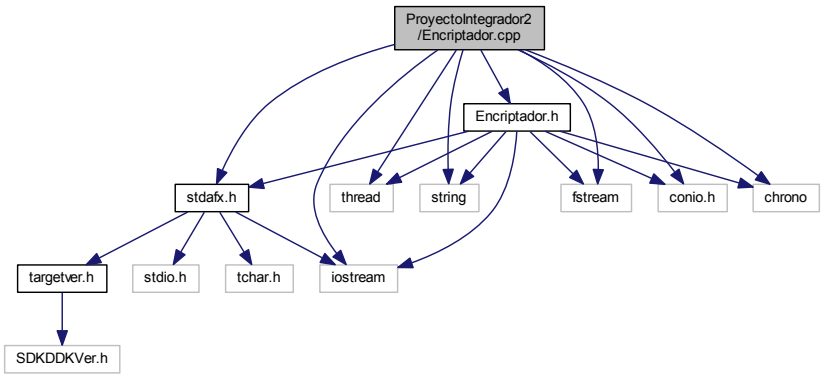
Classes

- class **Cuestionario**

5.7 ProyectoIntegrador2/Encriptador.cpp File Reference

```
#include "stdafx.h"
#include <thread>
#include <string>
#include <iostream>
#include <fstream>
#include <conio.h>
#include <chrono>
#include "Encriptador.h"
```

Include dependency graph for Encriptador.cpp:



Functions

- bool **validarAcceso** ()

5.7.1 Function Documentation

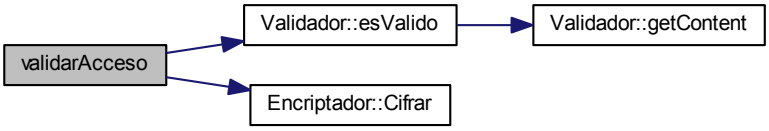
5.7.1.1 bool validarAcceso ( )

Definition at line 110 of file Encriptador.cpp.

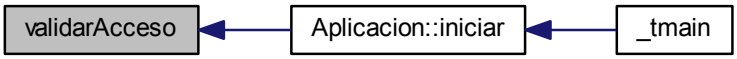
```
111 {
112     Encriptador crypt("RSeCDf23g4ihjlk6auvMLxpQUmAEBwzyFGontJHIqKr7 Ts8ON9PXVWZ0Y15bcd", "
    abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 ");
113     Validador valid("clave.bin");
114     string clave = "pablito clavo un clavito en la calva de un calvito";
115     //cout<<"Buscando el archivo"<<endl;
116     //if(valid.encontrado)cout<<"Archivo encontrado en " << valid.path<<endl;
117     //cout<<crypt.Cifrar(clave)<<endl;
118     bool validacion =valid.esValido(crypt.Cifrar(clave));
119     if (validacion)
120     {
121         //      Acceso=true;
122         //cout<<"Acceso Permitido"<<endl;
123         return true;
124     }
125     else
126     {
127         //      Acceso=false;
128         cout<<"Acceso Denegado"<<endl;
```

```
129
130         return false;
131
132     }
133     //cout<<"Contenido del archivo: "<<valid.getContent()<<endl;
134     //  cout<<"Contenido del archivo decifrado: " << crypt.Decifrar(valid.content)<<endl;
135     //  cout<<"Clave correcta: " << clave<<endl;
136 }
```

Here is the call graph for this function:



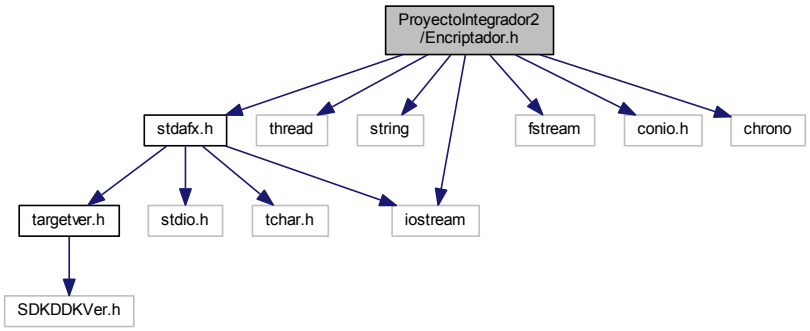
Here is the caller graph for this function:



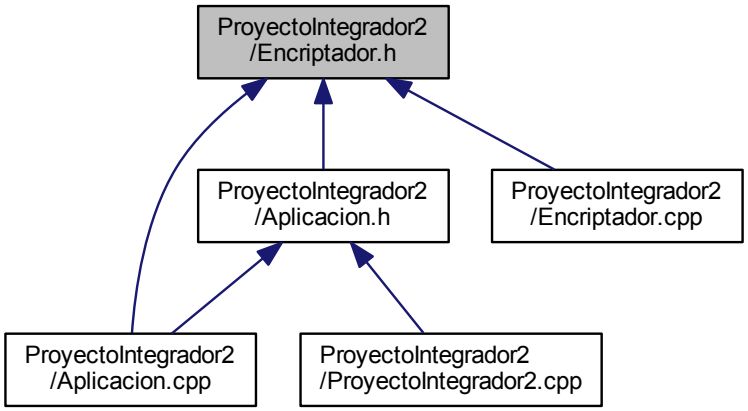
5.8 ProyectoIntegrador2/Encriptador.h File Reference

```
#include "stdafx.h"
#include <thread>
#include <string>
#include <iostream>
#include <fstream>
#include <conio.h>
#include <chrono>
```

Include dependency graph for Encriptador.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Encriptador**
- class **Validador**

Functions

- bool **validarAcceso** ()

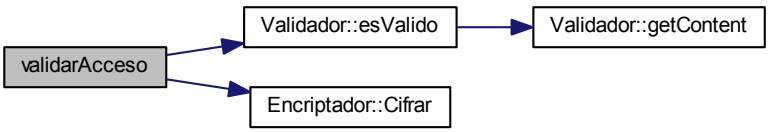
5.8.1 Function Documentation

5.8.1.1 bool validarAcceso ( )

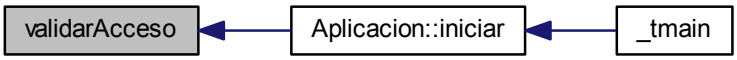
Definition at line 110 of file Encriptador.cpp.

```
111 {
112     Encriptador crypt("RSeCDf23g4ihjlk6auvMLxpQUmAEBwzyFGontJHIqKr7 Ts8ON9PXVWZ0Y15bcd", "
    abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 ");
113     Validador valid("clave.bin");
114     string clave = "pablito clavo un clavito en la calva de un calvito";
115     //cout<<"Buscando el archivo"<<endl;
116     //if(valid.encontrado)cout<<"Archivo encontrado en " << valid.path<<endl;
117     //cout<<crypt.Cifrar(clave)<<endl;
118     bool validacion =valid.esValido(crypt.Cifrar(clave));
119     if (validacion)
120     {
121         //      Acceso=true;
122         //cout<<"Acceso Permitido"<<endl;
123         return true;
124     }
125     else
126     {
127         //      Acceso=false;
128         cout<<"Acceso Denegado"<<endl;
129 
130         return false;
131 
132     }
133     //cout<<"Contenido del archivo: "<<valid.getContent()<<endl;
134     //  cout<<"Contenido del archivo decifrado: " << crypt.Decifrar(valid.content)<<endl;
135     //  cout<<"Clave correcta: " << clave<<endl;
136 }
```

Here is the call graph for this function:



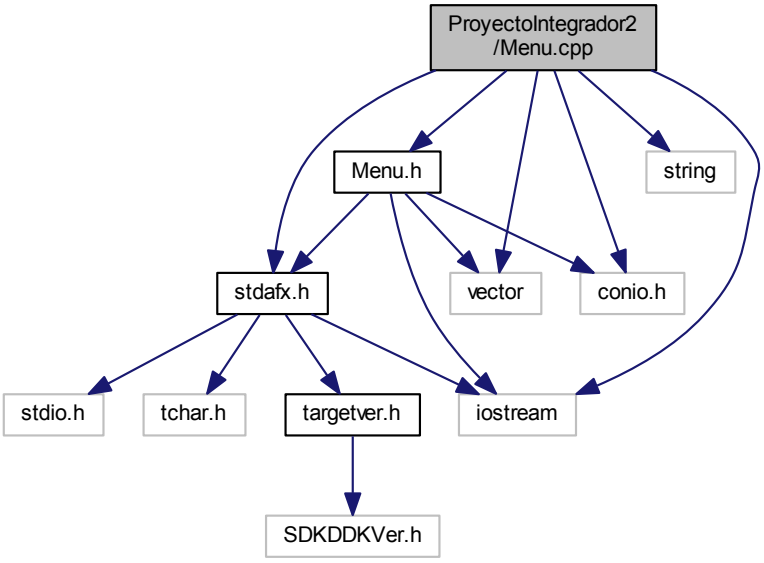
Here is the caller graph for this function:



5.9 ProyectoIntegrador2/Menu.cpp File Reference

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <conio.h>
#include <string>
#include "Menu.h"
```

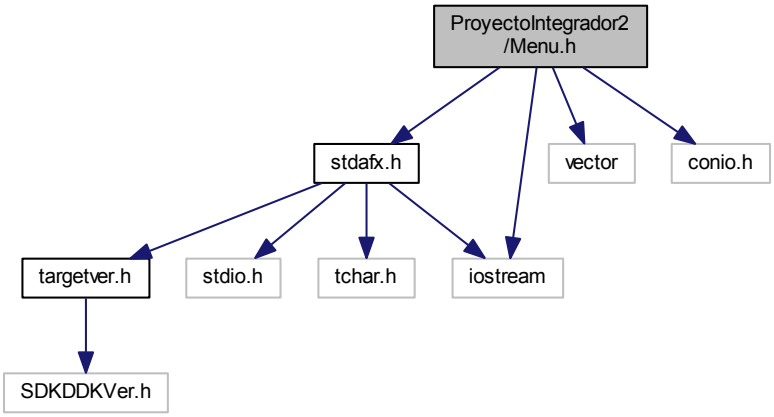
Include dependency graph for Menu.cpp:



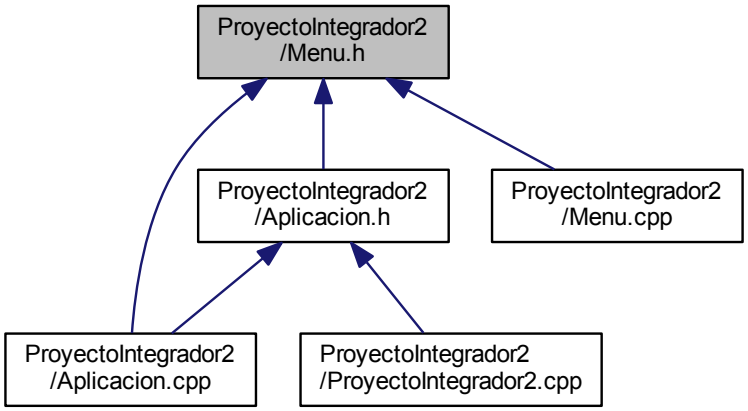
5.10 ProyectoIntegrador2/Menu.h File Reference

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <conio.h>
```

Include dependency graph for Menu.h:



This graph shows which files directly or indirectly include this file:



Classes

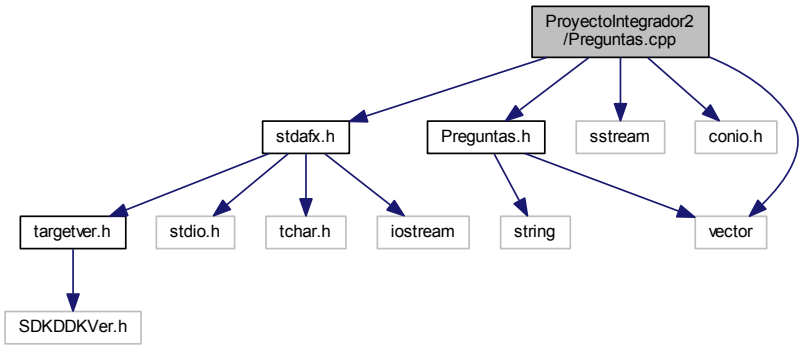
- class **Menu**

5.11 ProyectoIntegrador2/Preguntas.cpp File Reference

```
#include "stdafx.h"
```

```
#include "Preguntas.h"
#include <sstream>
#include <conio.h>
#include <vector>
```

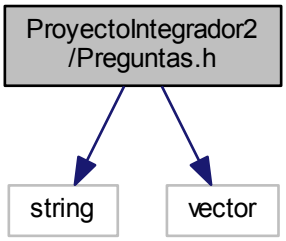
Include dependency graph for Preguntas.cpp:



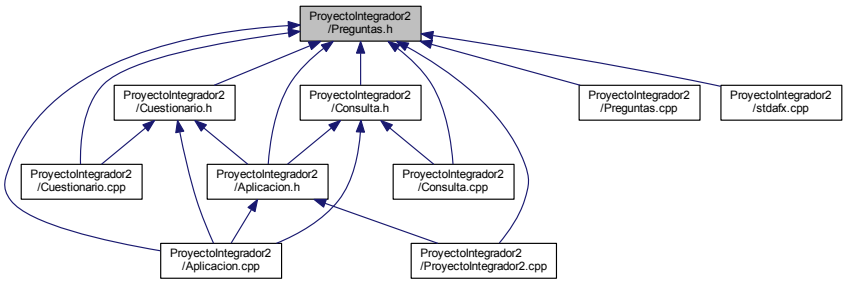
5.12 ProyectoIntegrador2/Preguntas.h File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for Preguntas.h:



This graph shows which files directly or indirectly include this file:



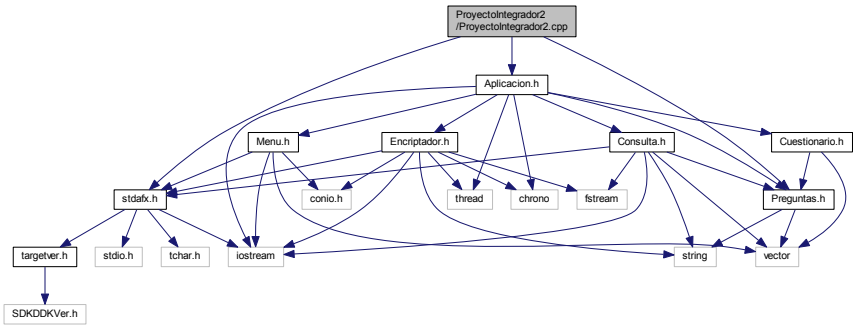
Classes

- class **Pregunta**
- class **Pregunta5a10**
- class **PreguntaOpcMult**

5.13 ProyectoIntegrador2/ProyectoIntegrador2.cpp File Reference

```
#include "stdafx.h"
#include "Preguntas.h"
#include "Aplicacion.h"
```

Include dependency graph for ProyectoIntegrador2.cpp:



Functions

- int **_tmain** (int argc, _TCHAR ∗argv[])

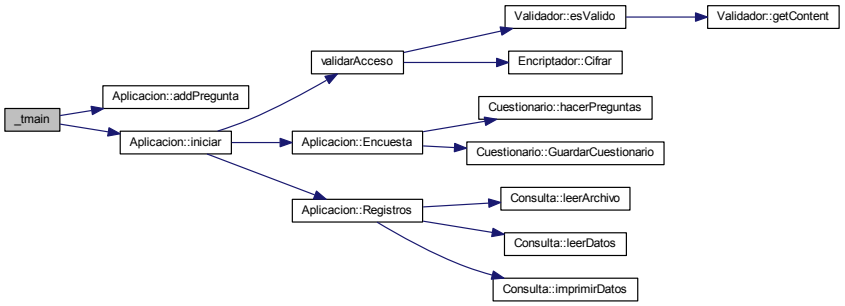
5.13.1 Function Documentation

5.13.1.1 int _tmain (int *argc*, _TCHAR * *argv[]*)

Definition at line 10 of file ProyectoIntegrador2.cpp.

```
11 {
12     Aplicacion programa;
13     vector<Pregunta*> preguntas; //Se crea un vector de punteros del tipo Pregunta, para almacenar las
       preguntas de la encuesta
14     preguntas.push_back(new Pregunta("Biblioteca"));
15     preguntas.push_back(new PreguntaOpcMult("Seleccione una Opcion","Alumno Docente Investigador    Otros")
       );
16     preguntas.push_back(new Pregunta("Carrera"));
17     preguntas.push_back(new Pregunta("Semestre"));
18     preguntas.push_back(new Pregunta5a10("Instalaciones (Comodidad, Limpieza, Iluminacion, Espacios)"));
19     preguntas.push_back(new Pregunta("Que mejoras propone?"));
20     preguntas.push_back(new Pregunta5a10("Acervo Bibliografico y documental (Actualizado, Suficiente)"));
21     preguntas.push_back(new Pregunta("Que mejoras propone?"));
22     preguntas.push_back(new Pregunta5a10("Herramientas y servicios de informacion (Prestamo de
       computadoras, Consulta en el catalogo, Acceso a bases de datos)"));
23     preguntas.push_back(new Pregunta("Que mejoras propone?"));
24     preguntas.push_back(new Pregunta5a10("Servicios del Personal (Rapidez, Asesoria)"));
25     preguntas.push_back(new Pregunta("Que mejoras propone?"));
26     for (size_t i=0;i<preguntas.size();i++) {programa.addPregunta(preguntas[i]);} //En este ciclo se an
       todas las preguntas del vector al cuestionario
27     programa.iniciar();
28     return 0;
29 }
```

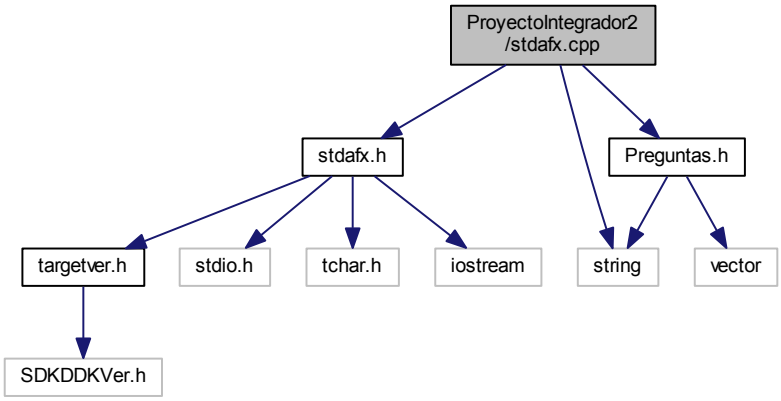
Here is the call graph for this function:



5.14 ProyectoIntegrador2/stdafx.cpp File Reference

```
#include "stdafx.h"
#include "Preguntas.h"
#include <string>
```

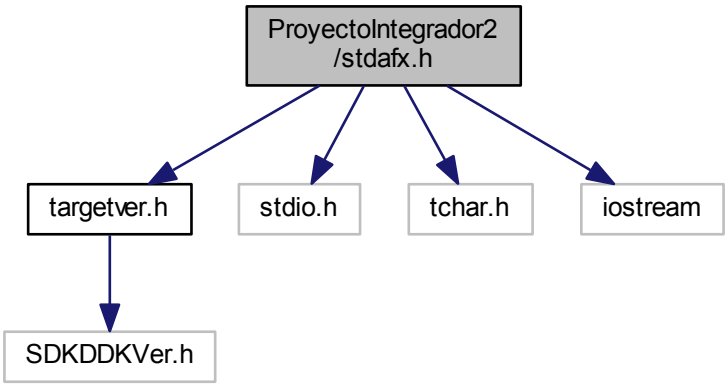
Include dependency graph for stdafx.cpp:



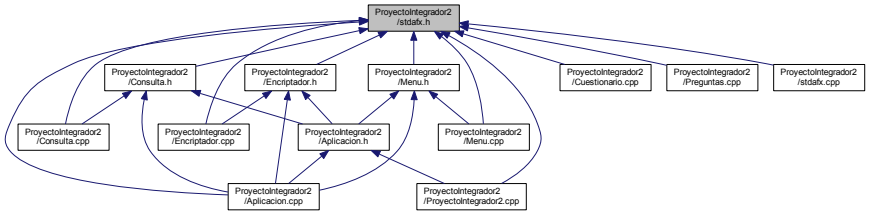
5.15 ProyectoIntegrador2/stdafx.h File Reference

```
#include "targetver.h"
#include <stdio.h>
#include <tchar.h>
#include <iostream>
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **_CRT_SECURE_NO_WARNINGS**

5.15.1 Macro Definition Documentation

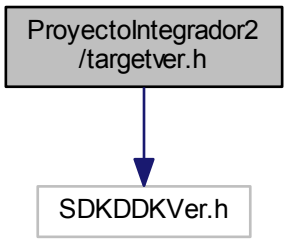
5.15.1.1 #define _CRT_SECURE_NO_WARNINGS

Definition at line 9 of file stdafx.h.

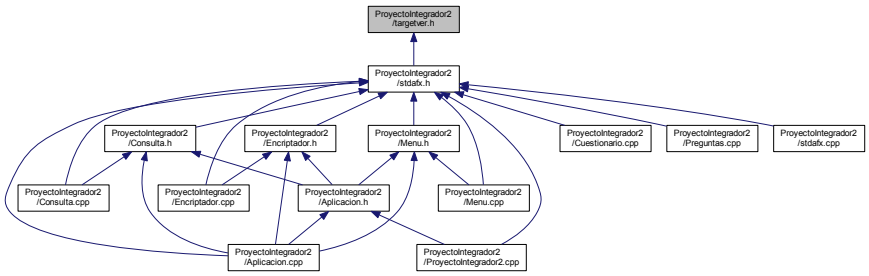
5.16 ProyectoIntegrador2/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

Include dependency graph for targetver.h:



This graph shows which files directly or indirectly include this file:



Index