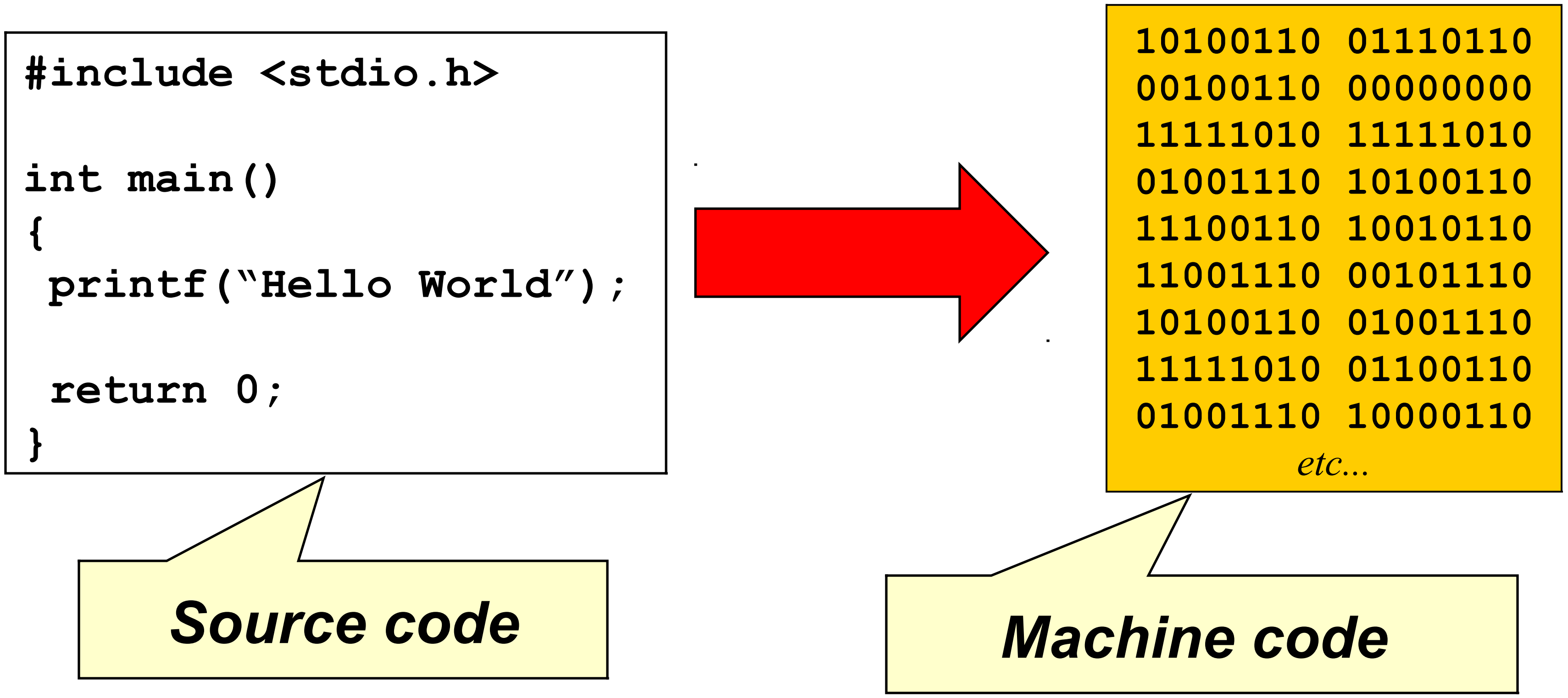
Introduction to C programming language

Department of Information System

SoICT, HUST

High level language



```
#include <stdio.h>

int main()
{
 printf(“Hello World”);

 return 0;
}
```

```
10100110 01110110
00100110 00000000
11111010 11111010
01001110 10100110
11100110 10010110
11001110 00101110
10100110 01001110
11111010 01100110
01001110 10000110
       etc...
```

- Compiler translates a program in high level programming language into machine language

Why C?

- Flexible language
  - Structured language
  - Low level activities possible
- Standard library exists, allowing portability
- It can produce lean and efficient code
- Wide availability on a variety of computers and widely used
- It is the foundation for other languages (C++, Java, Perl, awk)

History of C

- CPL Combined Programming Language (Barron et al., 1963)
- BCPL Basic CPL (Richards, 1969)
- B (Thompson, 1970)
- C K&R C (Ritchie, 1972)
- ANSI C American National Standards Institute C (X3J11, 1989)
- C99 (JTC1/SC22/WG14, ISO/IEC 9899, 1999)

The first C program

Hello World

Algorithm

output “Hello World!”

C program

```
#include <stdio.h>

int main()
{
  printf(“Hello World!”);

  return 0;
}
```

C Language Structure

- #include <stdio.h>
  - To declare using the standard I/O library. Other libraries: string, time, math…
- int main()
  - To declare the main() function. An C program must declare only one main() function. The first line in the main() will implement when the program starts.
- { … }
  - The syntax to open and close a block of codes.
- printf
  - the printf() function sends the output to standard output (monitor). This function will be taught in the next week.
- return 0;
  - Stop the program.

Syntax of C programs

- A C program is written using:
  - Keywords: reserved words for specific meaning in a program, e.g., main, if, do, while, …
  - User's names: names defined by user to specify a variable, a function, etc. in a program.
  - Specific characters: to represent expressions in a program and make the program have structure, for example:
    - Create a block of instructions {}
    - Create a string " "

Keywords of C

- Flow control (6) – `if, else, return, switch, case, default`
- Loops (5) – `for, do, while, break, continue`
- Common types (5) – `int, float, double, char, void`
- structures (3) – `struct, typedef, union`
- Counting and sizing things (2) – `enum, sizeof`
- Rare but still useful types (7) – `extern, signed, unsigned, long, short, static, const`
- Evil keywords which we avoid (1) – `goto`
- Weirdies (3) – `auto, register, volatile`

Common characters used in a program

- {…} create a block of instructions
- "…" create a string to display
- /* … */ create a block of comment in the program
- ; End of an instruction
- other characters for formulas such as +, -, *, /, (), …

Identifiers

- When declare a variable or a procedure, we need to identify it
- Principles:
  - Only use alphabetic letters, numbers, underscore _ character to name an identify
  - Identify must begin with an alphabetic letter
  - Upper case and lower case are different
- Which identities are illegal:
  - tong, 2k, trung binh, %totnghiep

Example of writing a program

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

Create the main fuction for the program

```
int main()
{




   return 0;
}
```

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```c
#include <stdio.h>

int main()
{




   return 0;
}
```

Declare the standard input/output library

Example …

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{



   return 0;
}
```

Comment

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{
   int dem;



   return 0;
}
```

Declare a variable

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```c
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{
   int dem;
   dem = 0;



   return 0;
}
```

Assign a value to dem

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```c
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{
   int dem;
   dem = 0;
   while ( dem < 10 )
   {


   }
   return 0;
}
```

Creare a loop

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{
   int dem;
   dem = 0;
   while ( dem < 10 )
   {
       printf(“%d\n”, dem);

   }
   return 0;
}
```

Example ...

print number from 0 to 9

```
dem = 0
while (dem < 10)
do
{
    output dem
    dem = dem + 1
}
```

```c
#include <stdio.h>

/* In tu 0 toi 9 */
int main()
{
   int dem;
   dem = 0;
   while ( dem < 10 )
   {
       printf("%d\n", dem);
       dem = dem + 1;
   }
   return 0;
}
```

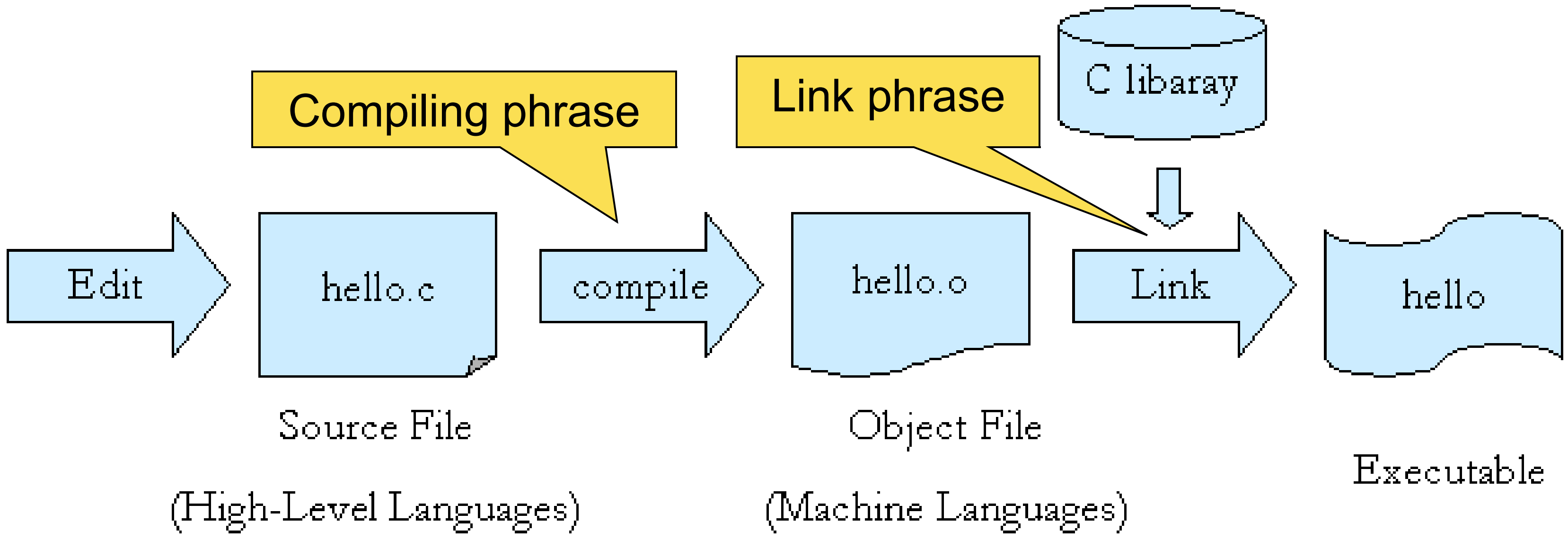
What does this program do?

```
#include <stdio.h>

int main(){
  float num;

  printf("Enter a number: ");
  scanf("%f", &num);

  if ( num < 0 )  {
     printf("%f is negative", num);
  }   else   {
     printf("%f is positive", num);
  }

  return 0;
}
```

Algorithm of the program

```
/* Find the sign of a number */
input num
if (num < 0) then
{
    output “negative number”
}
else
{
    output “positive number”
}
```

Compile a C program



Error can appear at compiling phrase or link phrase

Compiler

- To translate a program, we need a compiler, for example: gcc
- The compiler of C always supports parameter to perform two phrases of the compiling process. For example, gcc – c to do the compiling phrase, gcc –o to do the link phrase.
- If your program is written in only one file, a single Unix command can help to make an executable program from the source code.
  - $ gcc -o *program-name filename*
  - Ex: $gcc -o hello hello.c

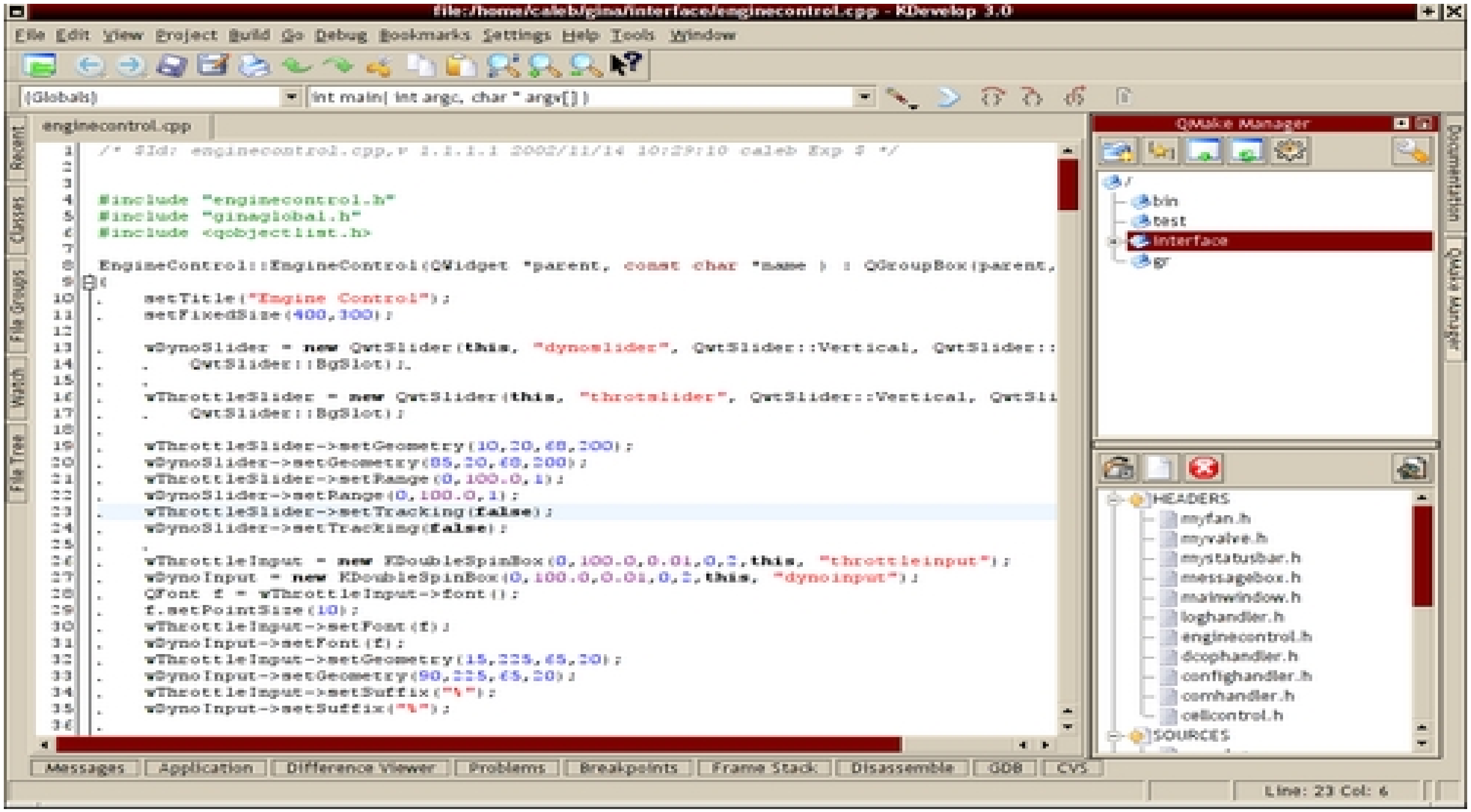
IDE: Integrated Development Environment

- Programming is a process that repeatly carries out operations: source code editing, compiling, debugging.
- These operations can carry out independently by different tools. For example, edit by emacs, compile by gcc.
- However, a more convenient way is to integrate all programming tools to an unique environment to support the programming. This environment is called IDE.
- IDE - an environment of 3 in 1: editor, compiler, debugger

Products of IDE

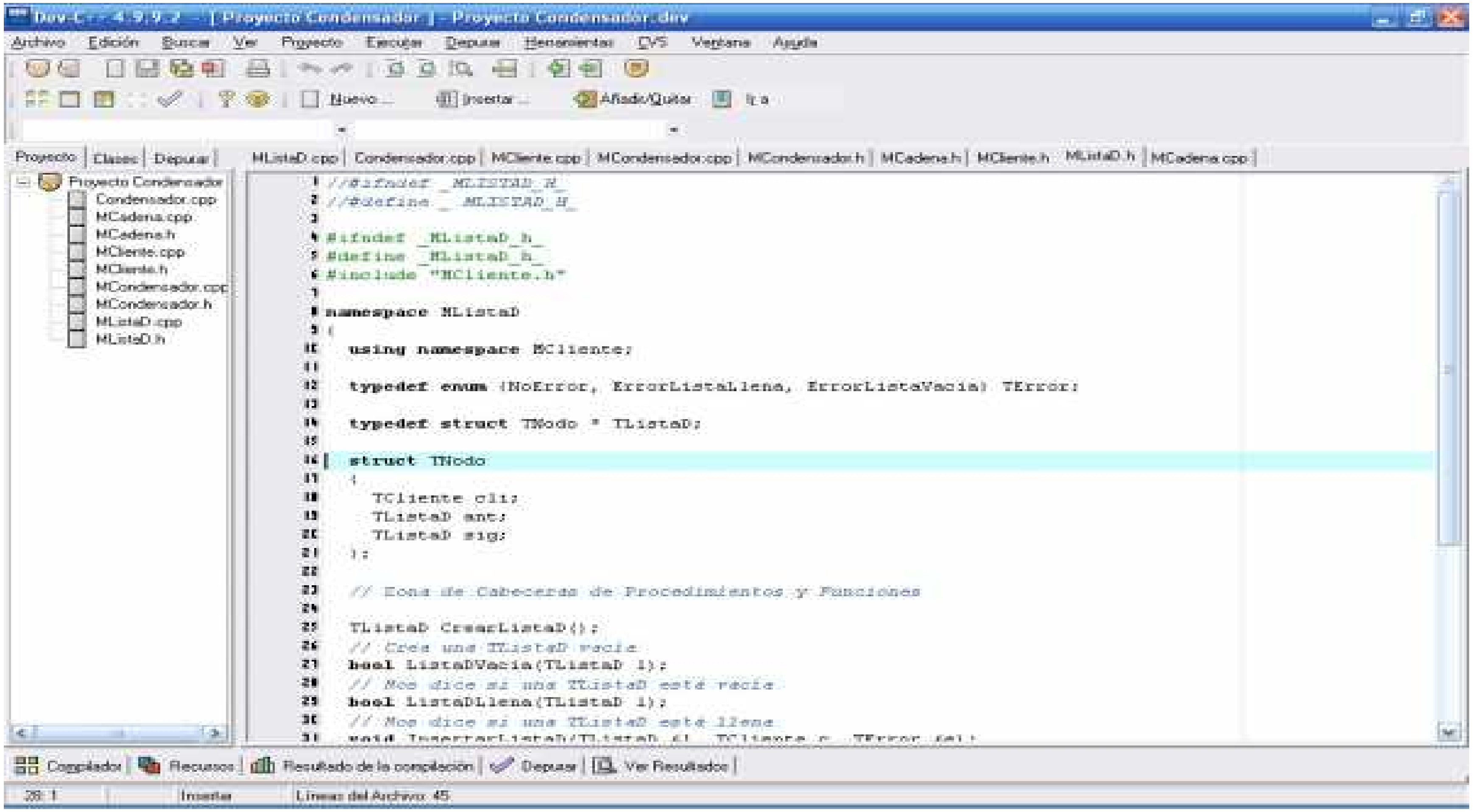
- On Linux:
  - KDevelop
- On Window:
  - Dev-C++,
  - Turbo C++,
  - Visual C++,
  - etc.

KDevelop



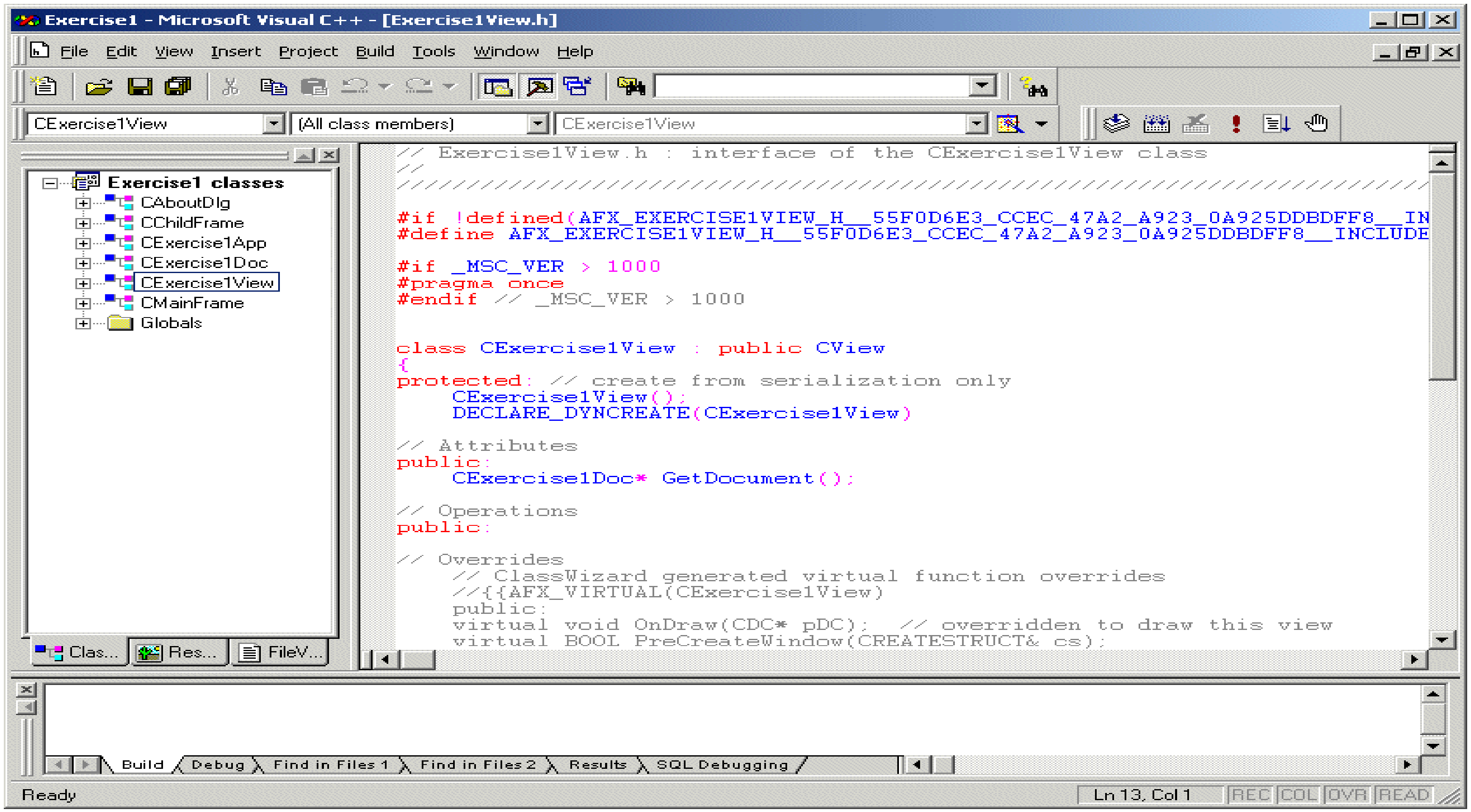
```
/* $Id: enginecontrol.cpp,v 1.1.1.1 2002/11/14 10:29:10 caleb Exp $ */


#include "enginecontrol.h"
#include "ginaglobal.h"
#include <qobjectlist.h>

EngineControl::EngineControl(QWidget *parent, const char *name ) : QGroupBox(parent,
{
.    setTitle("Engine Control");
.    setFixedSize(400,300);

.    wDynoSlider = new QwtSlider(this, "dynoslider", QwtSlider::Vertical, QwtSlider::
.    .    QwtSlider::BgSlot);.
.    .
.    wThrottleSlider = new QwtSlider(this, "throtslider", QwtSlider::Vertical, QwtSli
.    .    QwtSlider::BgSlot);
.
.    wThrottleSlider->setGeometry(10,20,60,200);
.    wDynoSlider->setGeometry(85,20,60,200);
.    wThrottleSlider->setRange(0,100.0,1);
.    wDynoSlider->setRange(0,100.0,1);
.    wThrottleSlider->setTracking(false);
.    wDynoSlider->setTracking(false);
.    .
.    wThrottleInput = new KDoubleSpinBox(0,100.0,0.01,0,2,this, "throttleinput");
.    wDynoInput = new KDoubleSpinBox(0,100.0,0.01,0,2,this, "dynoinput");
.    QFont f = wThrottleInput->font();
.    f.setPointSize(10);
.    wThrottleInput->setFont(f);
.    wDynoInput->setFont(f);
.    wThrottleInput->setGeometry(15,225,65,20);
.    wDynoInput->setGeometry(90,225,65,20);
.    wThrottleInput->setSuffix("%");
.    wDynoInput->setSuffix("%");
.
```

Dev-C++


Dev-C++ 4.9.9.2 - [Proyecto Condensador] - Proyecto Condensador.dev
Archivo Edición Buscar Ver Proyecto Ejecutar Depurar Herramientas CVS Ventana Ayuda
Nuevo Insertar Añadir/Quitar Ir a
Proyecto Clases Depurar
MListaD.cpp Condensador.cpp MCliente.cpp MCondensador.cpp MCondensador.h MCadena.h MCliente.h MListaD.h MCadena.cpp
Proyecto Condensador
Condensador.cpp
MCadena.cpp
MCadena.h
MCliente.cpp
MCliente.h
MCondensador.cpp
MCondensador.h
MListaD.cpp
MListaD.h
Compilador Recursos Resultado de la compilación Depurar Ver Resultados
28: 1 Insertar Líneas del Archivo: 45

```
//#ifndef _MLISTAD_H_
//#define _ MLISTAD_H_

#ifndef _MListaD_h_
#define _MListaD_h_
#include "MCliente.h"

namespace MListaD
{
  using namespace MCliente;

  typedef enum {NoError, ErrorListaLlena, ErrorListaVacia} TError;

  typedef struct TNodo * TListaD;

  struct TNodo
  {
    TCliente cli;
    TListaD ant;
    TListaD sig;
  };

  // Zona de Cabeceras de Procedimientos y Funciones

  TListaD CrearListaD();
  // Crea una TListaD vacia
  bool ListaDVacia(TListaD l);
  // Nos dice si una TListaD esta vacia
  bool ListaDLlena(TListaD l);
  // Nos dice si una TListaD esta llena
  void InsertarListaD(TListaD &l, TCliente c, TError &e);
```

Visual C++



```
// Exercise1View.h : interface of the CExercise1View class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_EXERCISE1VIEW_H__55F0D6E3_CCEC_47A2_A923_0A925DDBDFF8__IN
#define AFX_EXERCISE1VIEW_H__55F0D6E3_CCEC_47A2_A923_0A925DDBDFF8__INCLUDE

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CExercise1View : public CView
{
protected: // create from serialization only
	CExercise1View();
	DECLARE_DYNCREATE(CExercise1View)

// Attributes
public:
	CExercise1Doc* GetDocument();

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CExercise1View)
	public:
	virtual void OnDraw(CDC* pDC);  // overridden to draw this view
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
```