COMPILACIÓN DE UN PROGRAMA

Precompilación, Compilación y Enlace

Profesora: Mg. Silvia Vargas
Profesor: Mg. Miguel Guanira

El computador no entiende un programa como el que se muestra a continuación:

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    int a;
    a = 23*85;
    printf("Variable a = %d\n", a);
    return (EXIT_SUCCESS);
}
```

Para que lo pueda comprender, hay que “traducirlo” a un lenguaje que entienda, esto es al **“LENGUAJE MÁQUINA”**.

Al proceso de traducción se le denomina **“COMPILACIÓN”**

[2]

ETAPAS DE LA COMPILACIÓN:

- Precompilación
- Compilación propiamente dicha
- Enlace

[3]

PRECOMPILACIÓN

[4]

Como su nombre lo indica es un proceso previo a la compilación.

A continuación explicaremos este proceso

[5]

Regla general en el lenguaje C:

“Todo identificador debe ser declarado antes de poder utilizarlo”

[6]

Si borramos en el código la línea 5, el programa no compilará porque la variable **“a”** no fue declarada.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char** argv) {
5      int a;
6      a = 23*85;
7      printf("Variable a = %d\n", a);
8      return (EXIT_SUCCESS);
9  }
```

Entonces, cómo es que el identificador **“printf”** se está utilizando sin antes haberlo declarado.

[7]

La respuesta está en el PREPROCESADOR.

En el lenguaje C existe una serie de instrucciones, todas empiezan con el símbolo #, estas son instrucciones del preprocesador.

La precompilación consiste entonces en ejecutar las instrucciones del preprocesador antes de empezar la traducción de código del programa.

[8]

En el código del programa observamos la instrucción:

```
1  #include <stdio.h>
```

Esa instrucción se ejecuta entes que se traduzca el programa, su función es la de buscar el archivo de textos **“stdio.h”**, extraer todo el código que se encuentra en él y lo coloca en remplazo de esta instrucción.

En el texto del archivo se encuentra la declaración del identificador **“printf”**, por eso no tenemos que hacerlo nosotros cada vez que empezamos un programa.

```
7      printf("Variable a = %d\n", a);
```

[9]

Al igual que printf, en ese archivo existe la declaración de muchos identificadores, que podemos emplear en cualquier momento en el programa, y que se declaran con solo una orden del preprocesador.

El hecho que podamos emplear elementos que no hemos implementado ni hemos declarado en el programa que estamos desarrollando se le conoce con el nombre de **“Reutilización de código”**.

Un desarrollador profesional de software debe dominar y aplicar este último concepto en todos sus programas .

COMPILACIÓN PROPIAMENTE DICHA

La compilación propiamente dicha es la traducción del programa que hemos escrito en el lenguaje que entiende la máquina.

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    int a;
    a = 23*85;
    printf("Variable a = %d\n", a);
    return (EXIT_SUCCESS);
}
```

```
0110010111  1110011…
101010   111110   1011101…
…
```

PROGRAMA “FUENTE”

Archivo con extensión: c

PROGRAMA “OBJETO”

Archivo con extensión: o

A diferencia de lo que se puede pensar, la compilación no está completa, un programa objeto **NO** puede ejecutarse .

La razón de esto es que la traducción que se ha hecho es literal.

Ejemplo: Si a una persona que solo habla inglés usted le dice “construye una bomba” es claro que no le va a entender. Si lo traduce y le dice “build a bomb” la persona lo va a entender, pero es muy probable que no sepa cómo construir una bomba, tendríamos que entregarle un manual que le indique cómo hacerlo.

Ordenes como: `printf("Variable a = %d\n", a);`

La compilación hará que el computador entienda la orden pero no podrá ejecutarla porque le falta las instrucciones que le digan cómo hacerlo.

ENLACE

El enlace es la etapa final de la compilación.

Aquí el compilador creará un nuevo archivo agregándole al código del programa objeto, todo lo que requiera para ejecutarse .

Para ordenes como: `printf("Variable a = %d\n", a);`

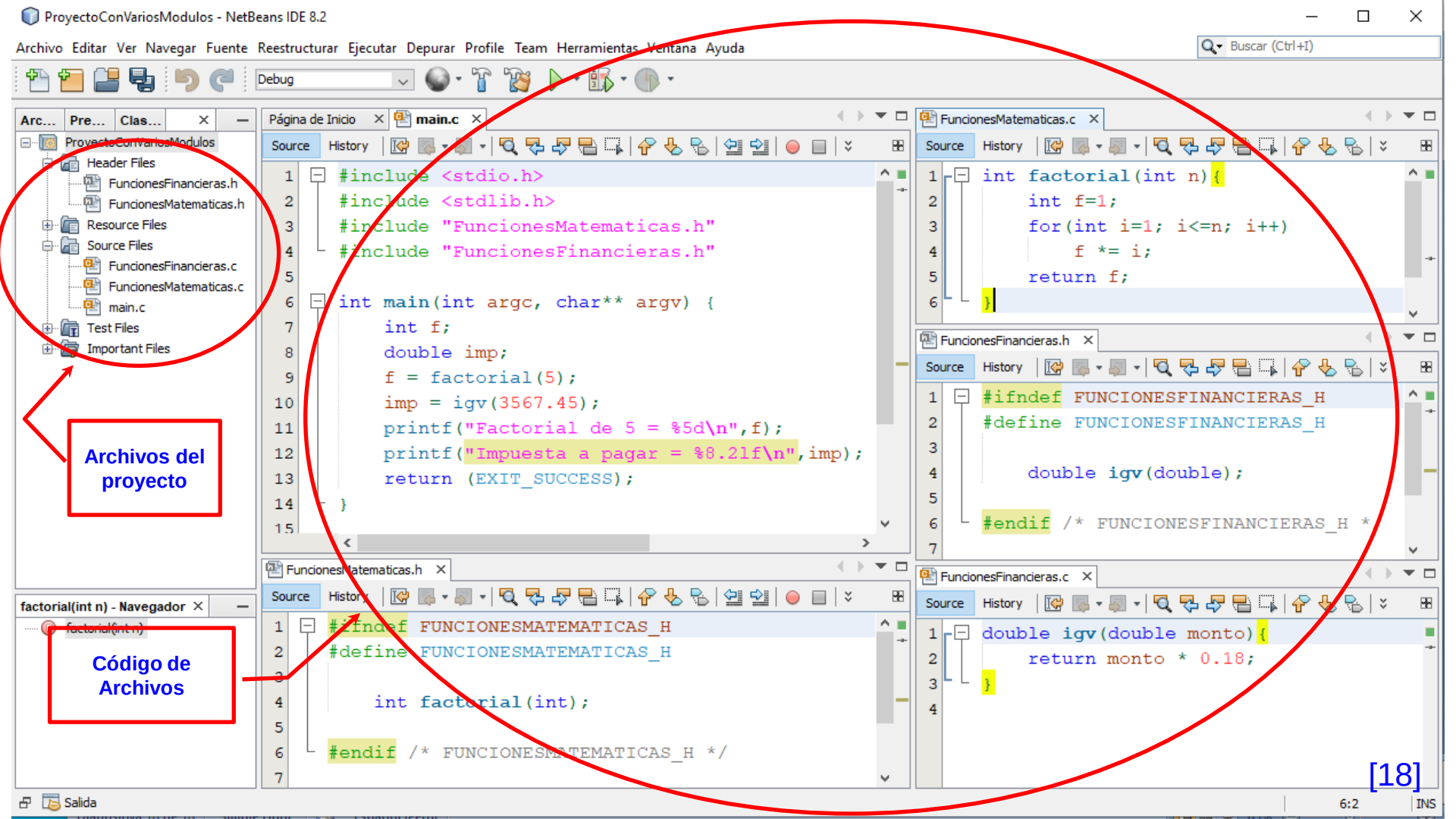
El compilador buscará el código en lenguaje máquina que permita ejecutar la orden y lo agregará al programa objeto.

También agregará las instrucciones necesarias para que el programa pueda ejecutarse independiente de cualquier otro programa (incluso del NetBeans).

COMPILACIÓN DE UN PROYECTO

NetBeans, el entorno de desarrollo que emplearemos en el curso, permite desarrollar proyectos. Esto quiere decir que los programas que realizaremos no los desarrollaremos en un solo archivo. El programa lo implementaremos en varios archivos, módulos o bibliotecas de funciones (más adelante se mostrará en detalle esto).

La razón de esto la dimos anteriormente: **“La posibilidad de reutilizar el código desarrollado en un proyecto”**.


ProyectoConVariosModulos - NetBeans IDE 8.2
Archivo Editar Ver Navegar Fuente Reestructurar Ejecutar Depurar Profile Team Herramientas Ventana Ayuda
Buscar (Ctrl+I)
Debug
ProyectoConVariosModulos
Header Files
FuncionesFinancieras.h
FuncionesMatematicas.h
Resource Files
Source Files
FuncionesFinancieras.c
FuncionesMatematicas.c
main.c
Test Files
Important Files
Archivos del proyecto
factorial(int n) - Navegador
Código de Archivos
Página de Inicio
main.c
#include <stdio.h>
#include <stdlib.h>
#include "FuncionesMatematicas.h"
#include "FuncionesFinancieras.h"
int main(int argc, char** argv) {
int f;
double imp;
f = factorial(5);
imp = igv(3567.45);
printf("Factorial de 5 = %5d\n",f);
printf("Impuesta a pagar = %8.2lf\n",imp);
return (EXIT_SUCCESS);
}
FuncionesMatematicas.c
int factorial(int n){
int f=1;
for(int i=1; i<=n; i++)
f *= i;
return f;
}
FuncionesFinancieras.h
#ifndef FUNCIONESFINANCIERAS_H
#define FUNCIONESFINANCIERAS_H
double igv(double);
#endif /* FUNCIONESFINANCIERAS_H *
FuncionesMatematicas.h
#ifndef FUNCIONESMATEMATICAS_H
#define FUNCIONESMATEMATICAS_H
int factorial(int);
#endif /* FUNCIONESMATEMATICAS_H */
FuncionesFinancieras.c
double igv(double monto){
return monto * 0.18;
}
Salida
6:2
INS

[18]

¿Cómo funciona la compilación es estos caso?

1 – Precompilación:

```c
#include <stdio.h>
#include <stdlib.h>
#include "FuncionesMatematicas.h"
#include "FuncionesFinancieras.h"

int main(int argc, char** argv) {
    int f;
    double imp;
    f = factorial(5);
    imp = igv(3567.45);
    printf("Factorial de 5 = %5d\n",f);
    printf("Impuesta a pagar = %8.2lf\n",imp);
    return (EXIT_SUCCESS);
}
```

FuncionesMatematicas.h

```c
#ifndef FUNCIONESMATEMATICAS_H
#define FUNCIONESMATEMATICAS_H

    int factorial(int);

#endif /* FUNCIONESMATEMATICAS_H */
```

FuncionesFinancieras.h

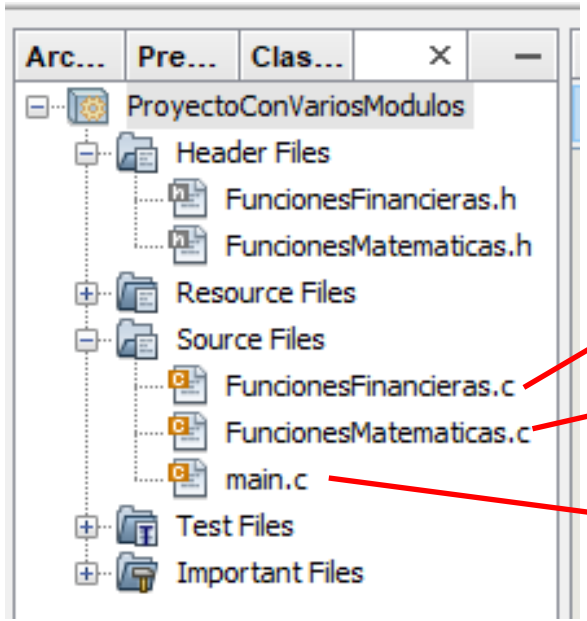
```c
#ifndef FUNCIONESFINANCIERAS_H
#define FUNCIONESFINANCIERAS_H

    double igv(double);

#endif /* FUNCIONESFINANCIERAS_H */
```

La precompilación reemplaza la orden #include por el código del archivo .h en cada uno de los archivos .c del proyecto.

2 – Compilación propiamente dicha:



FuncionesFinancieras.o

0110010111 1110011…
101010 111110 1011101…
…

FuncionesMatematicas.o

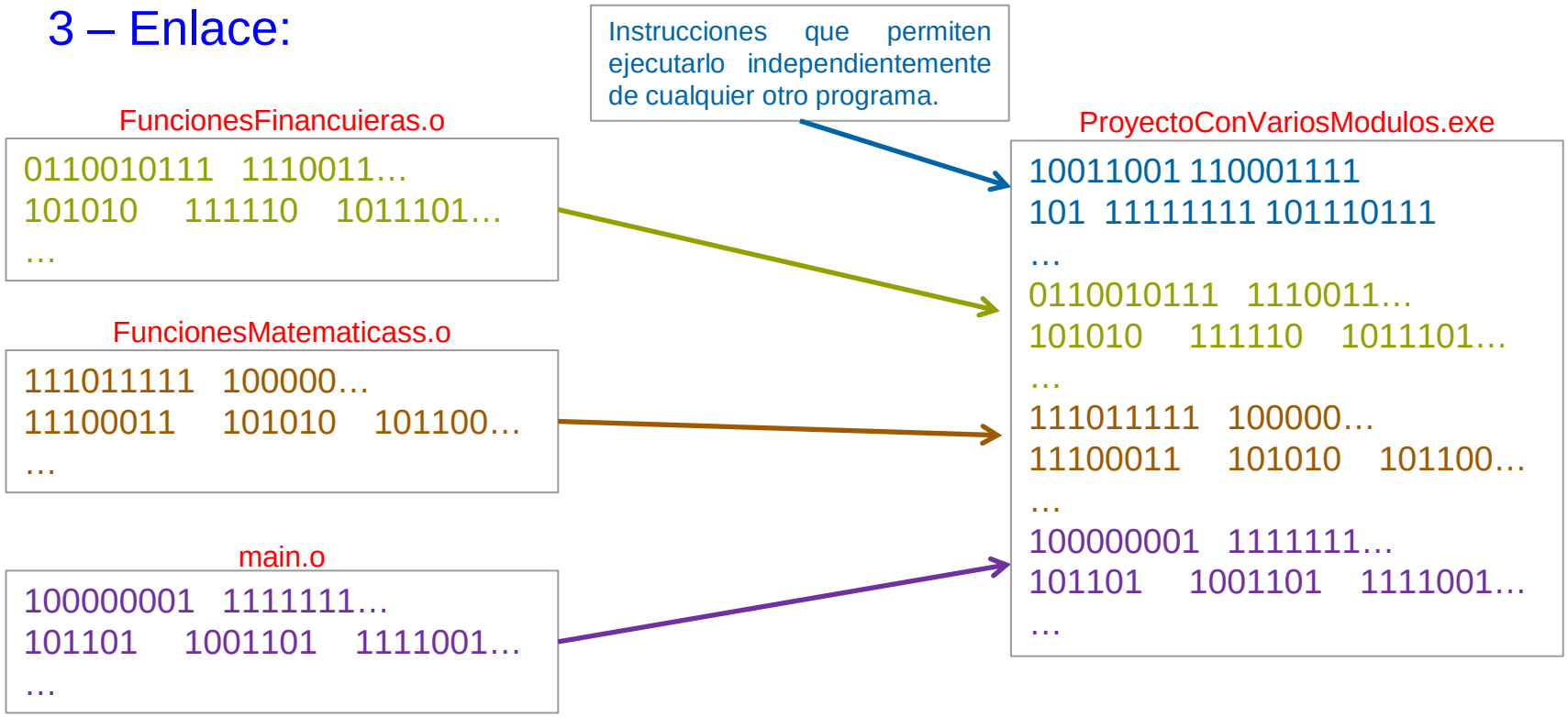
111011111 100000…
11100011 101010 101100…
…

main.o

100000001 1111111…
101101 1001101 1111001…
…

Cada archivo .c se traduce, independientemente de los otros, al lenguaje máquina y se guarda en un archivo .o

3 – Enlace:



Cada archivo .o se junta en uno solo, se le agrega a este último las instrucciones que permiten ejecutarlo independientemente y se crea con ellos el archivo ejecutable .exe.