****

Facultad de Ciencias Exactas y Naturales

Curso: Programación

Codigo: 0303158

Tema: Algoritmos, Diagramas, Pseudocódigo, Programas

Semestre: 2020-1

Medellín, 2020

**ALGORITMO, PSEUDOCÓDIGO, DIAGRAMA, PROGRAMA**

Algoritmo: secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas.

Cuando el problema debe resolverse por computador, la forma de expresar el algoritmo debe tener en cuenta el tipo de instrucciones que puede ejecutar el computador: entrada, salida, asignación, selección, etc.

Formas de expresar un algoritmo: pseudocódigo, diagrama, programa.

Ejemplo: multiplicación por un número entero de tres cifras.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 0 | 9 | x |  |
|  |  |  | 5 | 7 | 3 | y | = 100\*c+10\*d+u |
|  |  | 3 | 6 | 2 | 7 | P1 | = x\*u |
|  | 8 | 4 | 6 | 3 | 0 | P2 | = x\*d\*10 |
| 6 | 0 | 4 | 5 | 0 | 0 | P3 | = x\*c\*100 |
| 6 | 9 | 2 | 7 | 5 | 7 | z | = P1 + P2 + P3 = x\*y |

**1. Pseudocódigo**

Descripción del algoritmo con palabras del lenguaje corriente para referirse a las instrucciones, en vez de utilizar un lenguaje de programación específico.

Entrada: leer, pedir, introducir, capturar

Asignación: hacer, asignar, …

Salida: Escribir, mostrar, imprimir, …

Selección condicional: si… entonces…

**2. Diagrama**

Representación gráfica del algoritmo por medio de símbolos gráficos para referirse a las instrucciones, en vez de utilizar un lenguaje de programación específico.

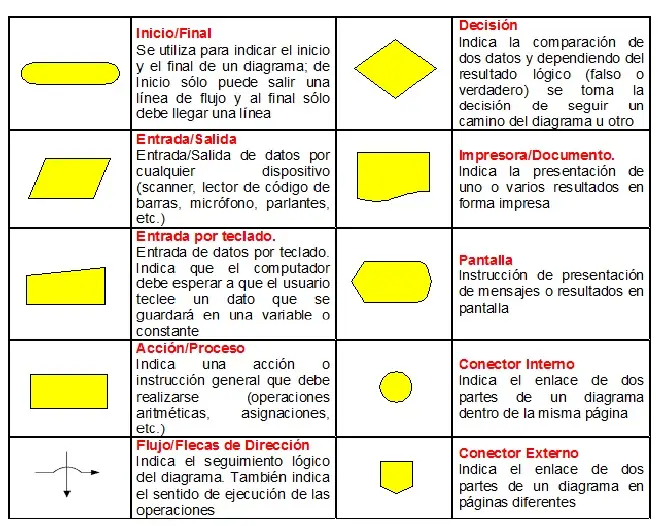


Figura 1. Símbolos para elaborar diagramas de flujo.

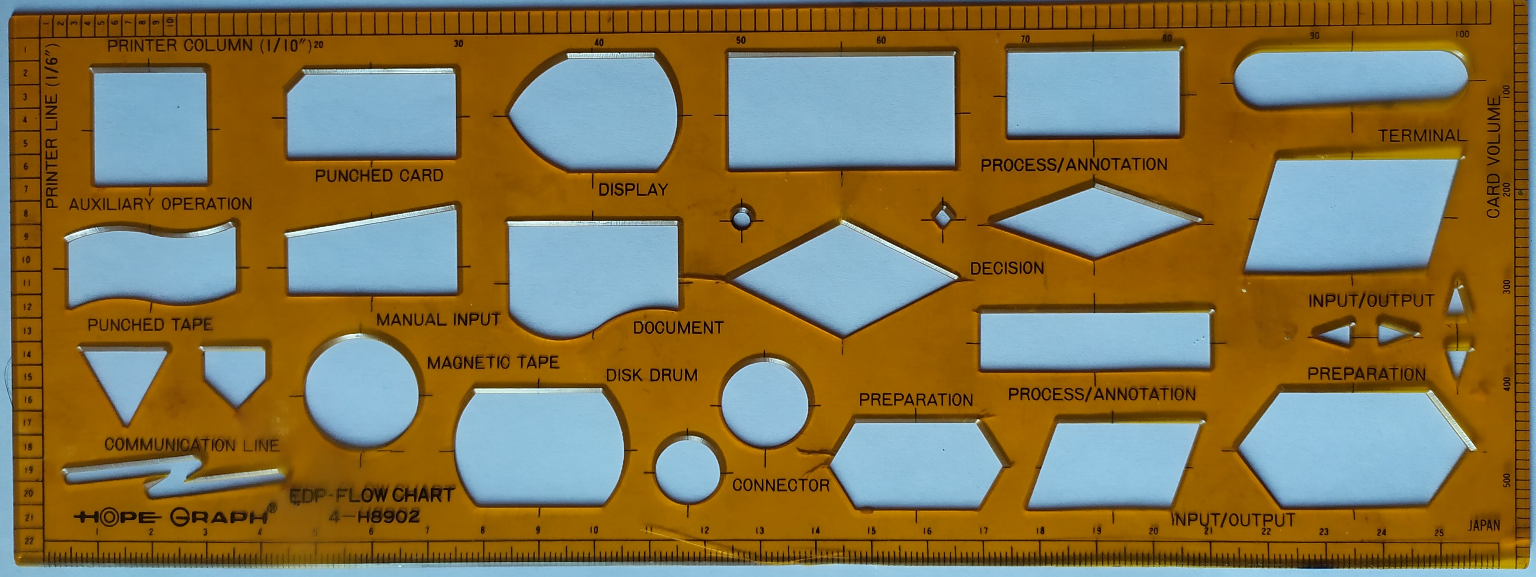


Figura 2. Plantilla para elaborar diagramas de flujo.

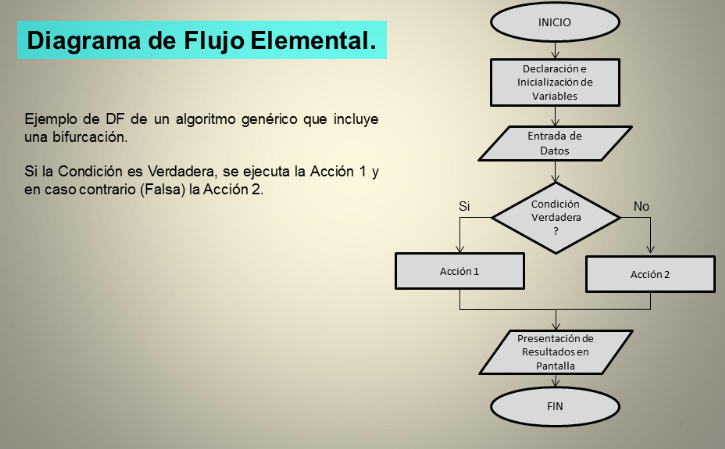


Figura 3. Ejemplo de un diagrama de flujo.

**3. Programas**

Archivos de texto plano que contienen secuencias de instrucciones en un lenguaje de programación de alto nivel (con palabras de un lenguaje conocido). Conocido también como Código Fuente. Mediante un proceso denominado **Compilación**, el codigo fuente se convierte en un programa ejecutable, es decir, una **Aplicación**.

Algunos entornos ejecutan las instrucciones directamente del código fuente y se conocen como **intérpretes**. Un intérprete no crea programas ejecutables. Octave y Matlab son intérpretes, aunque se consiguen compiladores de Matlab para empresas desarrolladoras de software (Matlab Compiler).

Otros entornos leen código Fuente y producen programas ejecutables (aplicaciones). El programa ejecutable resultante queda escrito en lenguaje de máquina (ceros y unos) y se puede ejecutar en ausencia del programa que lo creó.

**4. Entornos de programación**

Son aplicaciones que permiten desarrollar pseudocódigos, diagramas o programas, así como editarlos, probarlos y hasta desarrollar programas ejecutables, facilitando la tarea del programador.

Ejemplos: Matlab, DFD, PseInt, Code::Blocks.

DFD (Diagrama de Flujo de Datos) permite crear diagramas de flujo y ejecutarlos para verificar su correcto funcionamiento, aunque es bastante limitado por su incapacidad para mostrar correctamente el contenido de cada unidad del diagrama y su rigidez al ejecutar cada ‘instrucción’ en una ventana independiente. Se desarrolló una única versión en 1998.

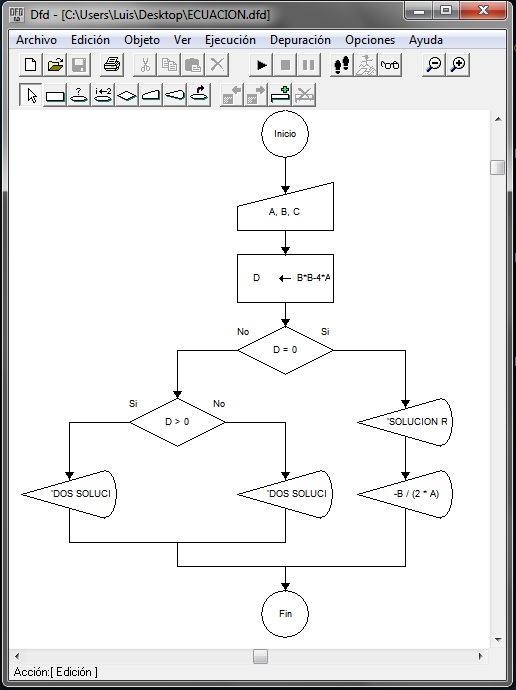


Figura 4. Ejemplo de un diagrama de flujo en DFD.

Pseint genera diagrama y código fuente a partir de pseudocódigo ejecutable. Tiene versiones desde 2003 hasta la fecha.

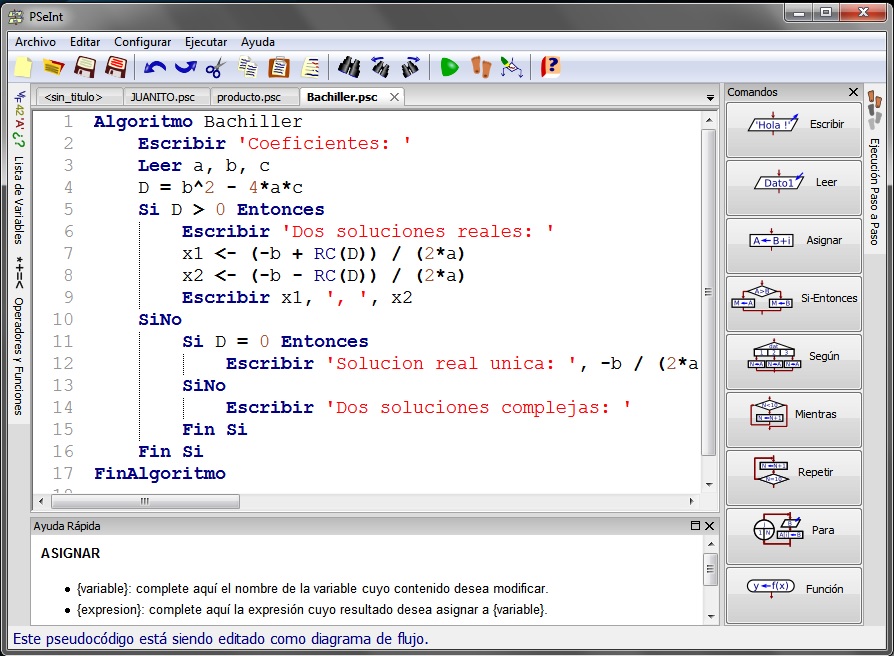


Figura 5. Pseudocódigo ejecutable en Pseint.

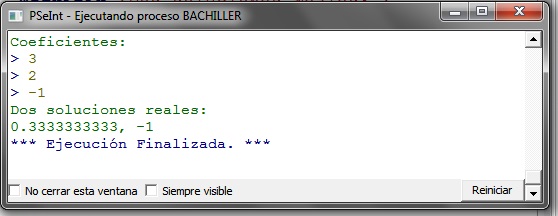


Figura 6. Ventana de ejecución en Pseint.

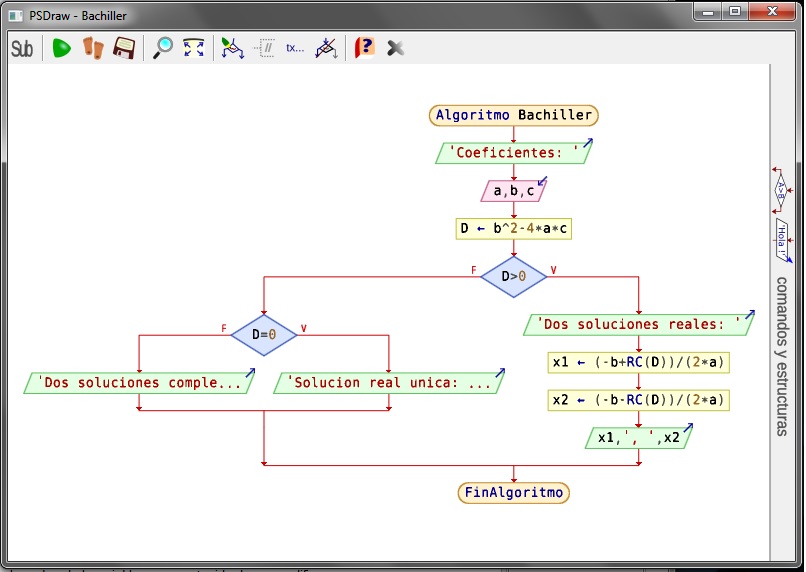


Figura 7. Diagrama generado en Pseint.

Code::Blocks es un editor que genera programas ejecutables a partir de un código fuente escrito en uno de los lenguajes para los cuales tenga instalado el correspondiente compilador, como C, C++, Pascal, Fortran, etc.

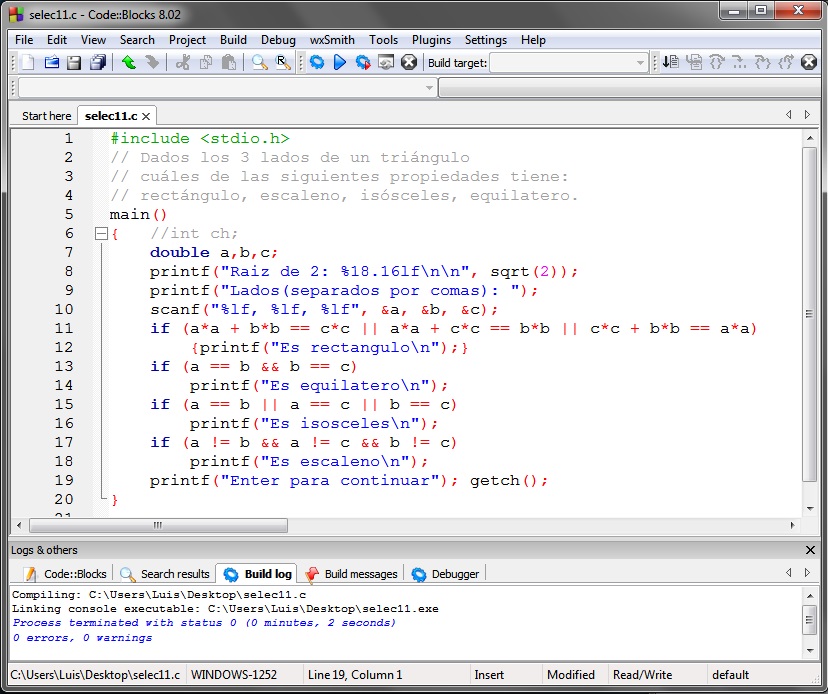


Figura 8. Código fuente en lenguaje C y ventana de compilación, en Code::Blocks.

**APENDICE**

**La instrucción fprintf, en Matlab**

Esta instrucción, conocida como instrucción de salida con línea de formato, permite un mayor control sobre la impresión que la instrucción **disp**.

Además de enviar resultados a la pantalla, puede crear archivos de texto como salida del programa.

Los archivos de texto creados con **fprintf** se pueden enviar a la impresora o pueden servir como entrada para otros programas (ver instrucción **fscanf**).

Sintaxis:

fprintf(<fid>, <cadena de formato>, <lista de expresiones>)

<fid>: identificador de archivo. Si se omite, el destino de la impresión es la pantalla

<cadena de formato>: contiene el texto que aparecerá en la impresión. Intercalados en el texto pueden incluirse valores de expresiones y saltos (controles de carro). Los valores de expresiones se expresan en diferentes formatos de salida para los que se utiliza el signo de porcentaje y una o más letras. Para los saltos se utiliza el backslash (\) y una o más letras.

<lista de expresiones>: es una lista de expresiones de cualquier tipo (aritméticas, lógicas o alfanuméricas) separadas por comas.

Controles de carro:

Permiten incluir controles de impresión tales como tabuladores y saltos de línea o parrafo. Se consideran caracteres especiales,para diferenciarlos del texto normal que aparece entre apóstrofos (comillas simples).

Tambien se consideran especiales los caracteres que tienen otro uso en Matlab, como los apóstrofos, el signo de porcentaje y el ‘backslash’.

| **Carácter especial** | **Representación** |
| --- | --- |
| Comilla simple | '' |
| Carácter de porcentaje | %% |
| Barra invertida | \\ |
| Alarma | \a |
| Retroceso | \b |
| Salto de impresión | \f |
| Nueva línea (salto de renglón) | \n |
| Retorno de carro (salto de párrafo) | \r |
| Tabulación horizontal | \t |
| Tabulación vertical | \v |

Figura 9. Controles de carro para fprintf.

Especificadores de formato o de conversión:

| **Tipo de valor** | **Conversión** | **Detalles** |
| --- | --- | --- |
| Entero, con signo | %d o %i | Base 10 |
| Entero, sin signo | %u | Base 10 |
| %o | Base 8 (octal) |
| %x | Base 16 (hexadecimal), letras en minúscula a–f |
| %X | Igual que %x, letras en mayúscula A–F |
| Número en punto flotante | %f | Notación en punto fijo (utilice un operador de precisión para especificar el número de dígitos después del punto decimal). |
| %e | Notación exponencial, como 3.141593e+00 (utilice un operador de precisión para especificar el número de dígitos después del punto decimal). |
| %E | Igual que %e, pero en mayúsculas, como 3.141593E+00 (utilice un operador de precisión para especificar el número de dígitos después del punto decimal). |
| %g | El más compacto entre %e y %f, sin ceros finales (utilice un operador de precisión para especificar el número de dígitos significativos.) |
| %G | El más compacto entre %E y %f, sin ceros finales (utilice un operador de precisión para especificar el número de dígitos significativos.) |
| Caracteres o cadenas | %c | Carácter único |
| %s | Cadena de caracteres. |

Figura 10. Especificadores de formato para fprintf.

Operadores de precisión

Dentro de un especificador de formato se pueden incluir operadores para indicar el espacio mínimo a ocupar por un valor, el número de cifras decimales y el tipo de alineación (izquierda o derecha).

Ejemplos (salidas a pantalla):

1. **Instrucciones:**

**fprintf('%6s %12s\r\n','x','exp(x)');**

**Salida:**

**x exp(x)**

1. **Instrucciones:**

**x = 0.5; fprintf('%6.2f %12.5f\r\n',x,exp(x));**

**Salida:**

**0.50 1.64872**

1. **Instrucciones:**

**A = pi/3;**

**fprintf('%16.3f %16.8f\r\n',pi,40\*sin(A));**

**fprintf('%16.3f %16.8f\r\n',pi+200,cos(A));**

**Salida:**

**3.142 34.64101615**

**203.142 0.50000000**

1. **Instrucciones:**

**A = pi/3;**

**fprintf('%-16.3f %-16.8f\r\n',pi,40\*sin(A));**

**fprintf('%-16.3f %-16.8f\r\n',pi+200,cos(A));**

**Salida:**

**3.142 34.64101615**

**203.142 0.50000000**

1. **Instrucciones:**

**A = 27; fprintf('%6.3f %12.8f\r\n',pi,A);**

**Salida:**

**3.142 27.00000000**

1. **Instrucciones:**

**pct = 37; filename = "foo.txt";**

**printf ("Processed %d%% of '%s'.\nPlease be patient.\n",**

**pct, filename);**

**Salida:**

**Processed 37% of 'foo.txt'.**

**Please be patient.**

1. **Instrucciones:**

**n1 = 'Abel'; a1 = 'Ruiz'; s1 = 327455;**

**n2 = 'Francisco'; a2 = 'Aristizabal'; s2 = 572;**

**fprintf('%15s%15s%12.2f\n',n1,a1,s1);**

**fprintf('%15s%15s%12.2f\n',n2,a2,s2);**

**fprintf('%15s%-15s%12.2f\n',n1,a1,s1);**

**fprintf('%15s%-15s%12.2f\n',n2,a2,s2);**

**fprintf('%-15s%-15s%-12.2f \n',n1,a1,s1);**

**fprintf('%-15s%-15s%-12.2f \n',n2,a2,s2);**

**fprintf('%-30s%12.2f\n',[n1,' ',a1],s1);**

**fprintf('%-30s%12.2f\n',[n2,' ',a2],s2);**

**fprintf('%30s%-12.2f\n',[n1,' ',a1],s1);**

**fprintf('%30s%-12.2f\n',[n2,' ',a2],s2);**

**Salida:**

**Abel Ruiz 327455.00**

**Francisco Aristizabal 572.00**

**AbelRuiz 327455.00**

**FranciscoAristizabal 572.00**

**Abel Ruiz 327455.00**

**Francisco Aristizabal 572.00**

**Abel Ruiz 327455.00**

**Francisco Aristizabal 572.00**

**Abel Ruiz327455.00**

**Francisco Aristizabal572.00**

1. **Instrucciones:**

**url='http://adn.co'; site = 'ADN';**

**fprintf('<a href = "%s">%s</a>\n',url,site);**

**Salida:**

**<a href = "http://adn.co">'ADN </a>**

Pregunta: ¿Con **disp** se puede manejar el número de decimales?

Ejemplos (salidas a un archivo de texto):

fprintf(<fid>, <cadena de formato>, <lista de expresiones>)

En los ejemplos de salida a pantalla, **fprintf** no llevaba el identificador de archivo. Éste es una variable que debe definirse previamente con una instrucción **fopen** y luego de enviar la información a imprimir en él, debe "cerrarse" con la instrucción **fclose**.

**Sintaxis de fopen:**

**<**fileID> = fopen(<filename>,<permission>)

**<**fileID>: es una variable que identificará el archivo durante el programa.

<filename>: es una cadena de caracteres con el nombre real del archivo.

<permission>: es una cadena de caracteres que indica para qué se va a abrir el archivo, como se indica en la siguiente tabla:

|  |  |
| --- | --- |
| **Código** | **Acción** |
| 'r' | Abra el archivo para leer. |
| 'w' | Abra o cree un nuevo archivo para escribir. Deseche los contenidos existentes, si los hubiere. |
| 'a' | Abra o cree un nuevo archivo para escribir. Anexar datos al final del archivo. |
| 'r+' | Abra el archivo para leer y escribir. |
| 'w+' | Abra o cree un nuevo archivo para leer y escribir. Deseche los contenidos existentes, si los hubiere. |
| 'a+' | Abra o cree un nuevo archivo para leer y escribir. Anexar datos al final del archivo. |
| 'A' | Abra el archivo para anexarlo sin purga automática del búfer de salida actual. |
| 'W' | Abra el archivo para escribir sin purga automática del búfer de salida actual. |

Figura 11. Permisos para abrir un archivo con fopen.

**Sintaxis de fclose:**

fclose(<fileID>)

**Ejemplo**: Escribir un script en Matlab para crear un archivo de texto con la colilla de pago de un trabajador.

Solución: Escribiremos el programa pago.m que pedirá los datos del trabajador y creará un archivo de texto llamado colilla.txt con la información correspondiente al pago.

El programa pago.m nos quedó así:

**% PAGO SEMANAL A UN TRABAJADOR**

**% Se paga por horas laboradas**

**% a un valor convenido por hora.**

**% Las horas que pasen de 40 son extras.**

**% La hora extra vale un 30% mas que la normal**

**% Por cada hijo se da un subsidio de $30.000**

**a = fopen('colilla.txt','a+');**

**nom = input('Nombre: ','s');**

**nro\_horas = input('Horas laboradas: ');**

**valor\_hora = input('Valor hora normal: ');**

**nro\_hijos = input('Numero de hijos: ');**

**if nro\_horas > 40**

**extras = nro\_horas -40;**

**normales = 40;**

**else**

**extras = 0;**

**normales = nro\_horas;**

**end**

**sal\_normal = normales\*valor\_hora;**

**sal\_extra = extras\*valor\_hora\*1.3;**

**subsidio = nro\_hijos\*3e4;**

**neto = sal\_normal + sal\_extra + subsidio;**

**fprintf(a, '\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n');**

**fprintf(a, 'Trabajador: %s\n', nom);**

**fprintf(a, '%25s %12d\n', 'Total horas laboradas:', nro\_horas);**

**fprintf(a, '%25s %12d\n', 'Horas normales:', normales);**

**fprintf(a, '%25s %12d\n', 'Horas extras:', extras);**

**fprintf(a, '%25s %12.2f\n', 'Salario normal:', sal\_normal);**

**fprintf(a, '%25s %12.2f\n', 'Salario extra:', sal\_extra);**

**fprintf(a, '%25s %12.2f\n', 'Subsidio familiar:', subsidio);**

**fprintf(a, '%25s %12.2f\n', 'Salario neto:', neto);**

**fprintf(a, '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n');**

**fclose(a);**

Figura 12. Codigo fuente del programa pago.m

El programa se ejecutó tres veces en Octave 3.2.4 de la siguiente manera:

Figura 13. Ejecucución del programa pago.m.

octave-3.2.4.exe:60> pago

Nombre: Adelaida Nieto

Horas laboradas: 35

Valor hora normal: 2000

Numero de hijos: 3

octave-3.2.4.exe:61> pago

Nombre: Pedro Ramirez

Horas laboradas: 48

Valor hora normal: 1800

Numero de hijos: 4

octave-3.2.4.exe:62> pago

Nombre: Manuela Rodriguez

Horas laboradas: 50

Valor hora normal: 3000

Numero de hijos: 0

octave-3.2.4.exe:63>

Se creó el archivo colilla.txt en el mismo directorio donde se guardan los programas de esta versión de Octave:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trabajador: Adelaida Nieto

Total horas laboradas: 35

Horas normales: 35

Horas extras: 0

Salario normal: 70000.00

Salario extra: 0.00

Subsidio familiar: 90000.00

Salario neto: 160000.00

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trabajador: Pedro Ramirez

Total horas laboradas: 48

Horas normales: 40

Horas extras: 8

Salario normal: 72000.00

Salario extra: 18720.00

Subsidio familiar: 120000.00

Salario neto: 210720.00

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trabajador: Manuela Rodriguez

Total horas laboradas: 50

Horas normales: 40

Horas extras: 10

Salario normal: 120000.00

Salario extra: 39000.00

Subsidio familiar: 0.00

Salario neto: 159000.00

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Figura 14. Contenido del archivo colilla.txt.