

PIPEH PSeudo Interprete 1.1

Contenido

1. Utilizacion del Entorno

2. Sintaxis del PseudoCodigo

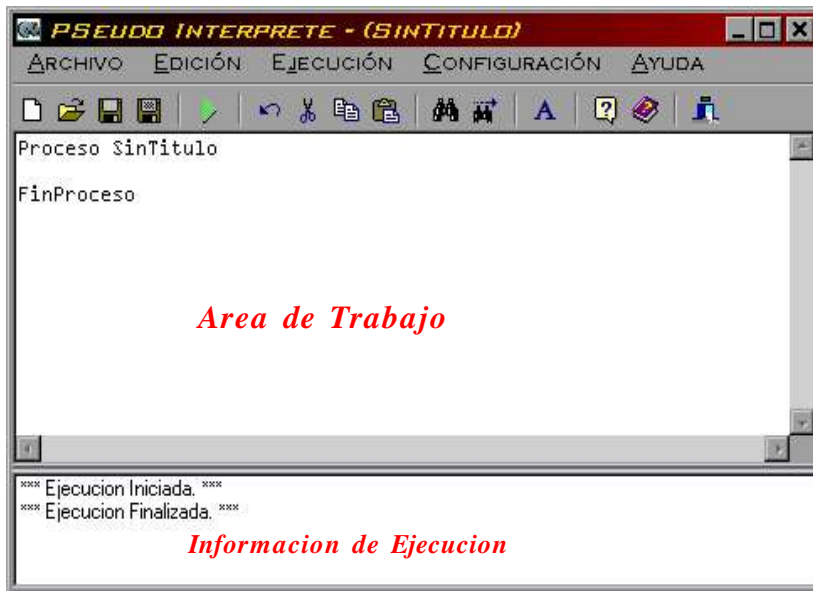
3. Mensajes de Error

4. Acerca de...



1. Utilizacion del entorno

Elementos de la Ventana



<- Barra de Menus

<- Barra de Acceso Rapido

Area de Trabajo

Informacion de Ejecucion

Menu Archivo



Nuevo: Crea un nuevo algoritmo y escribe el prototipo basico en el area de trabajo.

Abrir: Abre un algoritmo desde un archivo en formato txt o psc (texto ascii).

Guardar: Guarda el algoritmo actual con el nombre actual reemplazando al archivo existente.

Guardar Como: Guarda el algoritmo actual con un nuevo nombre.

Salir: Cierra la aplicacion.

Los elementos **1, 2** y **3** (entre salir y guardar como) son los tres ultimos archivos trabajados. Si el programa se utiliza por primera vez, estos no aparecen en el menu, sino que se agregaran cuando se abran o guarden dichos algoritmos.

Menu Edicion



Deshacer: Deshace el ultimo cambio realizado al algoritmo. Solo tiene un nivel.

Cortar, Copiar y **Pegar:** Interactuan entre el texto del area de trabajo y el portapapeles de windows.

Seleccionar Todo: Selecciona todo el algoritmo.

Buscar: Localiza una cadena dentro del algoritmo. No distingue entre mayusculas y minusculas.

Buscar Siguiente: Repite la ultima busqueda realizada comenzando desde la ubicacion del cursor.

Menu Ejecucion

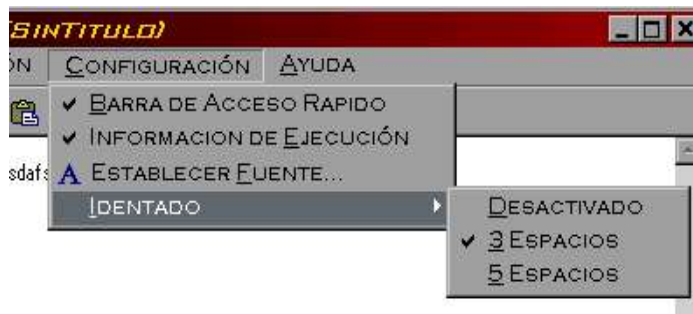


Iniciar: Ejecuta el algoritmo en una nueva ventana de consola.

Si el algoritmo contiene errores de sintaxis, no sera ejecutado.

Cuando finaliza actualiza al area de Informacion de Ejecucion. La ejecucion puede interrumpirse en cualquier momento presionando Ctrl+Pause, aunque de esta forma no se obtiene informacion de ejecucion. Tampoco se obtiene la informacion de ejecucion si se cierra dicha ventana desde su boton cerrar (la x en la esquina superior derecha). Lo correcto es presionar Enter luego de recibir los mensajes de errores o al finalizar el programa.

Menu Configuracion



Barra de Acceso Rápido: Muestra u oculta la barra.

Informacion De Ejecucion: Muestra u oculta la informacion.

Establecer Fuente: Permite seleccionar las propiedades de la fuente con la que se mostrara el algoritmo en el area de trabajo.

Identado: Modifica el funcionamiento de las teclas Enter, BackSpace y Tab para facilitar el identado del codigo en el margen izquierdo.

Menu Ayuda



Manual de Usuario: Muestra el Manual Este Manual.

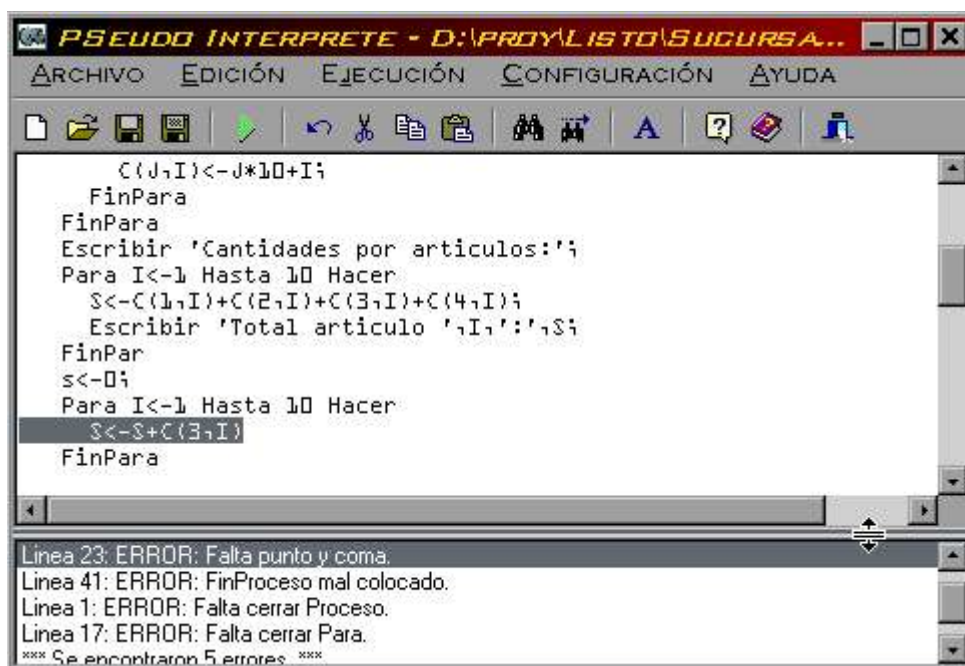
Manual de Referencia: Muestra la pagina de referencia. Dicha pagina contiene un resumen de la sintaxis de las instrucciones y los operadores de un algoritmo.

Acerca de...: Muestra informacion de la aplicacion.

Para poder visualizar los manuales es necesario tener instalado un visor de documentos PDF. En el CD se proporciona Adobe Acrobat Reader.

Informacion de Ejecucion

La Informacion de Ejecucion se actualiza cada vez que se ejecuta el algoritmo.



En ella aparecen los errores encontrados, ya sean de sintaxis (antes de ejecutarse) o errores en tiempo de ejecucion. Al hacer click sobre los mismos, automaticamente se seleccionara la linea que contiene dicho error en el area de trabajo.

Si la ejecucion se interrumpe de forma anormal (presionando Ctrl+Pause o finalizando el proceso mediante windows), dicha informacion no puede ser actualizada.

A partir de la version 1.1 se puede redimensionar esta area mediante el splitter ubicado entre dicha area y el area de trabajo. Notar el cursor del mouse como una doble flecha, manteniendo el boton izquierdo apretado podemos cambiar las dimensiones.

Problemas con el Entorno

Debido a que la comunicacion entre el entorno y el interprete propiamente dicho se realiza a travez de archivos temporales, es posible que no funcione correctamente si windows no se encuentra en "c:\windows" Para solucionar este problema, edite el archivo PseInt.ini que se encuentra en el directorio de instalacion. Si no lo encuentra, debe crearlo en la carpeta junto a los ejecutables. En dicho archivo agregue en cualquier lugar una linea como la siguiente:

```
WinDir=E:\windows\
```

En este ejemplo, Windows se encuentra en la carpeta "c:\windows. Notese que la ruta termina en "\". Esto siempre debe ser asi. No seria correcto "e:\windows".

Otro posible problema, es que al ejecutar la aplicacion un mensaje indique que no se encontro el interprete. Debe asegurarse de ejecutar VPSeInt.exe desde el directorio donde instalo la aplicacion. Si intento actualizar el interprete, consulte la documentacion adjunta.

Si esto no soluciona su problema, escriba a zaskar@hotmail.com y describa con detalle el error en cuestion.

2. Sintaxis del PseudoCodigo

Consideraciones Previas

Forma general de un algoritmo escrito en pseudocodigo:

```
Proceso <nombre>
    <instruccion 1>;
    <instruccion 2>;
    <instruccion 3>;
    ...
FinProceso
```

Todos los algoritmos deben comenzar con la palabra *Proceso* y a continuacion el nombre del mismo y deben finalizar con la palabra *FinProceso*. No puede haber instrucciones fuera del proceso, aunque si comentarios.

Las estructuras no secuenciales puede anidarse. Es decir, pueden contener otras adentro, pero la estructura contenida debe comenzar y finalizar dentro de la contenedora.

Se pueden introducir comentarios luego de una instruccion o en lineas separadas mediante el uso de la doble barra (//). Todo lo que precede a // no sera tomado en cuenta al interpretar el algoritmo.

Acerca de constantes e identificadores

- ◆ Los identificadores deben constar solo de letras y numeros, comenzando siempre con una letra.
- ◆ Las constantes de tipo caracter se escriben entre comas (').
- ◆ En las constantes numericas, el punto (.) es el separador decimal.
- ◆ Las constates logicas son Verdadero y Falso.

Asignacion

```
<variable> <- <expresion> ;
```

Primero evalua la expresion de la derecha y luego asigna el resultado a la variable de la izquierda. Deben coincidir en tipo.

Ejemplos:

```
var<-1; // asigna 1 a var
```

```
Acumulador<-Acumulador +1; // incrementa en 1 el  
acumulador
```

Entrada

```
Leer <variable1> , <variable2> , ... , <variableN> ;
```

Lee desde el ambiente (en este caso el teclado) los valores y los asigna a la o las variables respectivamente. Puede leer una o mas variables.

Ejemplos:

```
Leer Cantidad;
```

```
Leer Valor1,Valor2,Valor3;
```

Salida

```
Escribir <expr1> , <expr2> , ... , <exprN> ;
```

Devuelve al exterior (en este caso la pantalla) los resultados de las expresiones, o los contenidos de las variables. Tambien puede tener uno o mas parametros separados por comas.

Ejemplos:

```
Escribir 'Ingrese el nombre:' ;
```

```
Escribir 'Resultado:' , x*2 ;
```

Dimensionamiento

```
Dimesion <identificador1> (<max1>, ..., <maxN>) , ... ;
```

Define un arreglo. Es decir, la cantidad de dimensiones y el valor maximo de cada una de ellas. Tambien pueden declararse mas de un arreglo en la misma linea separandolos por comas. La cantidad de dimensiones puede ser una o mas, y la maxima cantidad de elementos debe ser una expresion numerica positiva. Es necesario definir un arreglo antes de utilizarlo. Se pueden definir mas de un arreglo en una misma instruccion, separandolos con una coma (,).

Ejemplos:

```
Dimension Alumnos ( 10 ) ;  
Leer Alumno (1);
```

```
Dimension Tabla ( 10 , 5 , 3 ) , Resultados ( 5 ) ;  
...  
Escribir Tabla (1,3,2);
```

Condicional

```
Si <condicion>  
  Entonces  
    ...  
  Sino  
    ...  
FinSi
```

Evalua la condicion y ejecuta las acciones que correspondan; *Entonces* si es verdadera, *Sino* si es falsa. Siempre debe estar presente *Entonces*, pero puede no aparecer *Sino*, y asi no hacer nada en caso de que la condicion sea falsa. La condicion puede ser una variable o una expresion logica de tipo logica.

Ejemplos:

```
Condicion<- A>B;  
Si Condicion  
  Entonces Escribir 'Si';
```

```
Sino Escribir 'No';
FinSi

Si Cantidad > 10
Entonces
    Escribir 'Demasiados articulos.';
    Escribir 'Solo se consideraran los 10 primeros.';
FinSi
```

Seleccion Multiple

```
Segun <variable> Hacer
    A: <...>;
    B,C: <...>;
    ...
    De Otro Modo: <...>;
FinSegun
```

Permite elegir entre dos o mas posibles grupos de acciones. Evalua el contenido de la variable y selecciona el camino a seguir. La variable debe ser de tipo numerico. Las opciones se delimitan por los dos puntos al final. Si un grupo de acciones se debe ejecutar en dos o mas casos, los valores se pueden poner separados por comas en la misma linea. La opcion final puede ser *De Otro Modo*, y se ejecuta si la variable no coincide con ninguna de las anteriores.

Ejemplo:

```
Leer a;
Segun a Hacer
    1: Escribir 1;
    2: Escribir 2;
    3,4: Escribir '3 o 4';
    De Otro Modo:
        Escribir 'No esta entre 1 y 4';
FinSegun
```

Mientras Hacer

```
Mientras <condicion> Hacer
    <...>
FinMientras
```

Ejecuta un grupo de acciones mientras la condicion sea verdadera. Puede que no se ejecuten nunca, si al momento de entrar en la estructura la condicion ya es falsa. Debe contener dentro alguna accion que pueda modificar la condicion para salir.

Ejemplo:

```
Escribir 'Ingrese un numero o 0 para salir';
Leer a;
Mientras a<>0 Hacer
    Tot<-Tot+a;
    Escribir 'Ingrese otro numero o 0 para salir';
FinMientras
Escribir 'Total:',Tot;
```

Repetir Hasta Que

```
Repetir
    <...>
Hasta Que <condicion>
```

Ejecuta un grupo de acciones hasta que la condicion sea verdadera. Como la condicion se evalua al final, siempre las acciones seran ejecutadas al menos una vez. Debe contener dentro alguna accion que pueda modificar la condicion para salir.

Ejemplo:

```
Repetir
    Tot<-Tot+a;
    Escribir 'Ingrese un numero o 0 para salir';
Hasta Que a=0
Escribir 'Total:',Tot;
```

Para

```
Para <I> <- <VI> Hasta <VF> ( Con Paso <P> ) Hacer
    <...>
FinPara
```

Funciones Matematicas

<i>Funcion</i>	<i>Significado</i>
RC(X)	Raiz Cuadrada de X
ABS(X)	Valor Absoluto de X
LN(X)	Logaritmo Natural de X
EXP(X)	Funcion Exponencial de X
SEN(X)	Seno de X
COS(X)	Coseno de X
ATAN(X)	Arcontangente de X
TRUNC(X)	Parte entera de X
REDON(X)	Entero mas cercano a X

La funcion raiz cuadrada no debe recibir un argumento negativo.
La funcion exponencial no debe recibir un argumento menor o igual a cero.

3. Mensajes de Error

Lista de posibles errores del algoritmo por orden alfabetico:

Asignacion incompleta.

La causa de este error esta en la ausencia de alguna de las dos partes de una asignacion, (la variable o la expresion a asignar).

Ejemplos:

```
Para <-3 hasta 5 hacer
a<-i
```

Caracter no valido.

Se encontro un caracter no que no existe en el pseudo codigo.
Ejemplos: &#\^°... Estos solo pueden aparecer dentro de constantes de tipo caracter.

Constante o Identificador no valido.

Los identificadores deben comenzar siempre con una letra y pueden continuar solo con letras o numeros. No pueden ser palabras reservadas. Tambien lo puede causar una constante incorrecta.

Ejemplos:

```
Escribir 1erVar;      // No puede empezar con numero
Tot<-Tot+ Primer_Var; // _ no es valido
Escribir 3.4.5;      // el nro tiene dos puntos
                    decimales
Escribir Para;      // Para es palabra reservada
```

Demasidos Parametros

Ocurre cuando una instruccion que admite solo un paramtro recibe mas de uno.

Ejemplo:

```
a<-1,3; // 1 y 3 son dos parametros
```

Division por cero

Una expresion matematica no debe intentar dividir por cero.

Ejemplo:

```
a<-0;
...
Escribir 7/a; // 7/a equivale a 7/0 y es imposible.
```

Entonces mal colocado.

Entonces siempre debe preceder a un Si, de lo contrario esta mal colocado.

Falta cerrar comillas.

Este error es ocasionado por una constante de tipo caracter incorrecta. Se deben delimitar con comillas simples (') tanto el inicio como el final de la misma.

Ejemplo:

```
Escribir 'Total:',Tot;
```

Falta cerrar parentesis.

Cuando en alguna expresion se abre un parentesis que luego no se cierra. Se deben cerrar todos los parentesis que se abren.

Ejemplo:

Escribir $(3+4/(2*a))$;

Falta cerrar Si/Para/Mientras/Repetir/Segun

Todas las estructuras deben tener un comienzo y un final dentro del algoritmo. Puede faltar la palabra FinSi/FinPara/FinMientras/Hasta Que/Fin Segun o puede que este mal ubicada.

Falta condicion.

Las estructuras de control repetitivas (Mientras, Repetir ... Hasta que) y las condicionales (Si y Segun) requieren las condiciones que determinan la salida del bucle.

Ejemplos:

Si
Hasta Que
Segun Hacer
Mientras

Falta Hacer.

Las instrucciones Para, Mientras y Segun, deben terminar con la palabra Hacer.

Ejemplos:

Para a<-1 hasta c
Segun a

Falta Hasta.

La instruccion para siempre debe contener Hasta <valor final>.

Ejemplos:

Para a<-1 hacer

Falta nombre de proceso.

Todos los procesos deben comenzar con la palabra Proceso y a continuacion un nombre valido para el mismo.

Ejemplo:

```
Proceso // aqui se coloca el nombre
```

Falta punto y coma.

Las acciones secuenciales debem finalizar siempre en punto y coma (;).

Ejemplos:

```
Escribir 1
Leer a
Dimension b(10)
```

Faltan Parametros.

Se produce cuando se intenta realizar una accion sin los parametros minimos necesarios.

Ejemplos:

```
Escribir ; // falta expresion
Leer ; // falta identificador de variable
Dimension ; // falta identificador de arreglo
Para B // falta asignacion, y valor final
```

Faltan subindices.

En la declaracion de un arreglo, siempre deben indicarse las dimensiones y sus cantidades maximas de elementos.

Ejemplo:

```
Dimension C;
```

Operador incorrecto.

Se encontro un operador mal ubicado o desconocido. Tambien ocurre cuando se encuentran dos operadores seguidos sin una constante o variable en medio.

Ejemplos:

```
a<-5*/3;          // */ dos operadores seguidos
Escribir 5++2;    // ++ no es un operador
```

Parametro nulo.

El parametro no puede estar vacio.

Ejemplos:

```
Escribir a,,2; // el 2do parametro esta vacio
Leer b,; // el 2do parametro esta vacio
Escribir (); // () es un parametro nulo
```

Raiz de negativo.

Solo se trabaja con numeros dentro del campo de los reales, por lo que no se puede calcular la raiz cuadrada de un numero negativo.

Se cerraron parentesis demás.

Se produce cuando en alguna expresion matematica sobran parentesis.

Ejemplo:

```
Escribir 2+(2*a)-b);
```

Se esperaba <opcion>:.

Despues de iniciar la estructura de seleccion multiple (Segun).
Deben introducirse al menos una opcion, seguida por dos puntos (:) y
luego las acciones a realizar.

Ejemplos:

```
Segun a
  Escribir b;
```

Se esperaba asignacion.

La instruccion para siempre debe contener la asignacion que
inicialize la variable de control.

Ejemplo:

```
Para a hasta 5 hacer
```

Se esperaba Con Paso o fin de instruccion.

Cuando en una instruccion para se introduce un parametro (u otra
instruccion) no valido.

Ejemplos:

```
Para a<-1 hasta 3 leer a;
```

Se esperaba entonces.

El condicional siempre debe tener una salida por verdadero.

Ejemplos:

```
Si a=1
  Sino Escribir 2;
FinSi
// Para evitar este error
```

```
// se puede replantear la condicion:  
// Ejs: a<>1 ?(a=1)
```

Se esperaba espacio o coma.

Cuando se encuentran dos variables o constantes consecutivas sin ningun separados en medio.

Ejemplo:

```
Escribir C B; // falta , u operador  
Escribir '21' B; // falta , 'u operador  
Mientras a='Si'Hacer // Falta espacio
```

Se esperaba Proceso <nombre>.

Todos los procesos deben comenzar con la palabra Proceso y a continuacion un nombre valido para el mismo. Todas las acciones deben estar contenidas dentro del proceso.

Sino mal colocado.

Sino siempre debe estar dentro de un Si, de lo contrario esta mal colocado.

Subindices fuera de rango

Se produce cuando se llama a un elemento de un arreglo ubicado en una posicion fuera se las definidas en la instruccion dimension.

Ejemplo:

```
Dimension a(5);  
...  
Escribir a(6); // a solo tiene 5 elementos.
```

4. Acerca de...

PIPEH Pseudo Interprete v1.1 (b23102004)

Cualquier pregunta, sugerencia o comentario puede ser enviado a:
zaskar@hotmail.com



PseInt fue desarrollado como proyecto final para la asignatura Programación 1 de la carrera Ingeniería en Informática.

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Santa Fe – Argentina – Diciembre de 2003

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you

be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands ``show w'` and ``show c'` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ``show w'` and ``show c'` they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.