MANUAL DE USUARIO

Requerimientos

Sistema Operativo (Windows o Linux)

Node js, Angular

Entender los lenguajes JAVA, PYTHON Y CPP (para editar archivos en el IDE).

Los archivos deben de tener la extensión: .mlg

Descripción

El programa es un editro de código, el cual genera código de tres direcciones a partir del archivo de entrada que se este leyendo.

Ventana Principal

En esta ventana es donde se abre, o crean los archivos respetivos del

programa.

Opciones Disponibles Pestaña Archivo

Nuevo: Despliega la pestaña denominada Programa, que contendrá exclusivamente el código fuente del programa a compilar.

Abrir: Esta opción permite abrir un archivo .mlg, éste contiene el código fuente de un programa almacenado con anterioridad. El contenido de este archivo será mostrado en la pestaña denominada Programa. Esta opción es únicamente para los archivos con extensión .mlg.

Guardar: Guarda el contenido de la pestaña activa en ese momento, en la dirección que el usuario desee.

Guardar Como: Permite guardar el contenido de la pestaña activa en ese momento, con diferente nombre, pero con la extensión que le corresponda.

Salir: Finaliza la ejecución de la aplicación.

Pestaña Generar Código

Actualmente solo se cuenta con la opción de Código de Tres Direcciones, este genera el código del mismo en el editor de texto.

Pestaña de Reportes

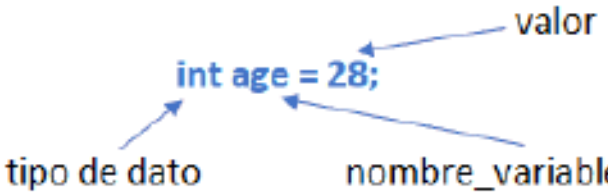
Actualmente solo se cuenta con el reporte de Errores, este se muestra en caso de que existan errores en el texto o código.

Descripción Lenguajes

Los lenguajes solo admiten las variables primitivas Integer, Float y Char.

JAVA

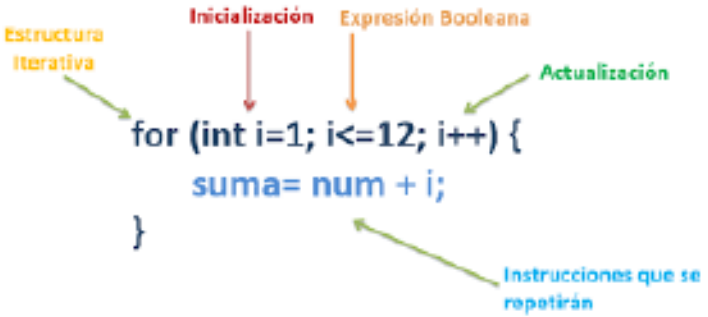
Declaración Variables



Bucles

```
int i = 0;
while (i < 10) {
    System.out.println("i: " + i);
    i++;
}
```

```
int i = 5;
do {
    System.out.println(i);
    i++;
} while (i <= 10);
```



Condicionales

```
if (condicion1){
    sentenciasA;
} else if (condicion2){
    sentenciasB;
} else if (condicion3){
    sentenciasC;
}
```

```
switch(expresion){

      case valor1:
           sentencias B1;
      break;

      case valor2:
           sentencias B2;
      break;

      case valor3:
           sentencias B3;
      break;

      ...

      [default:
           sentencias Bd;]

}
```

Lógicos

And =

&& Or =

|| Not = ¡

Diferente = ¡=

PY

Variables

$$x + 3 = 5$$

Bucles

```
i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

```
for num in range(0, 11, 2):
    print(num)
```

Condicionales

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

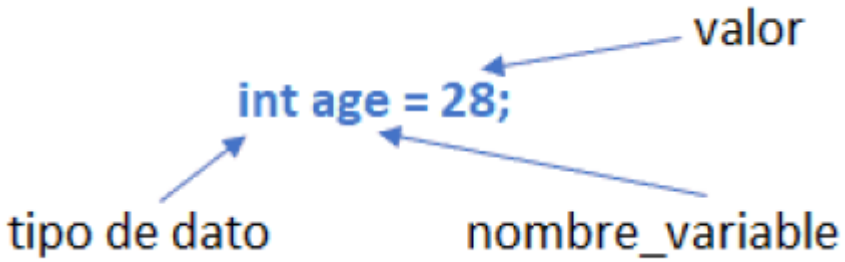
Lógicos And,

or, not

```
def evaluacion (nota):
    valoracion="aprobado"
    if nota<5:
        valoracion="suspenso"
        return valoracion
```

CPP

Variables



Bucles

Condicionales

```
if (condicion1){
     sentenciasA;
} else if (condicion2){
     sentenciasB;
} else if (condicion3){
     sentenciasC;
}
```

```
switch(expresion){

     case valor1:
          sentencias B1;
     break;

     case valor2:
          sentencias B2;
     break;

     case valor3:
          sentencias B3;
     break;

     ...

     [default:
          sentencias B4;]

}
```

Lógicos

And =

&& Or =

|| Not = ¡

Diferente = ¡=

Descripción del Archivo de Entrada

```
paquete com.paquete1     //Igual al de paquetes en java

%%PY
   Funciones y Procedimientos en Python
%%JAVA
   Clases en Java
%%PROGRAMA
Sección de librerías de C     //opcionales
#include "PY"            //incluye el código en Python
#include "PY.dir_a_paquete.archivo"//incluye el código en Python
definido en el archivo 'archivo'.
#include "JAVA.*"        //incluye todas las clases declaradas en
                         //la sección de Java
#include "JAVA.nombre"   //incluye   una   clase   en   especifico
                         //declarada en la sección de Java
#include "JAVA.dir_a_paquete.nombre"     //incluye    una    clase
                         existente en el paquete dir_a_paquete
#include "JAVA.dir_a_paquete.*"    //incluye   todas   las   clases
                         existente en el paquete dir_a_paquete
// no hay orden en las instrucciones de la seccion de librerias

Sección declaración de Constantes
Sección de Variables Globales

void main()
{
 // Programa principal
}
```

Los lenguajes que se utilizarán para conformar la sección de librerías son:

- Python
- Java

Mientras que el programa principal utilizará C, para cada sección del programa se deberá respetar la sintaxis de cada lenguaje de programación.

El area del programa principal puede o no contener código, al igual que la sección de python y de java.

- Expresiones aritméticas + - * / % ^ (potencia)
- mensajes a pantalla: se usará la palabra print y println tanto para python como en java, y la sintaxis es print(valores_separados_por_coma)
- solicitud de datos desde java a usuarios: intinput, floatinput, charinput, ej. vardestino = intinput()
- solicitud de datos desde python a usuarios: input ej. vardestino = input()
- ciclos (for i, while, do while), incluyendo continue y break.
- En python se usará el ciclo for con range: for x in range(int_exp)
- manejo de condiciones (if, switch)
- declaración de variables
- asignaciones variables
- Funciones
- Procedimientos
- Clases (Java)
- El paso de parametros cumple con Pass-by-object-reference
- La cadena es considerada como un tipo primitivo.
- Por simplicidad los metodos Python no pueden invocar a otros metodos python.
- Por simplicidad los metodos en clases de Java solo pueden invocar a metodos de la misma clase y de sus clases padre.

Además los únicos tipos primitivos que se manejan en estos lenguajes son entero, real, carácter, boolean y string.

NOTA:

Las declaraciones de funciones, métodos en python se manejarán de forma pública.

El programa principal que se maneja en sintaxis de C se especifica a continuación:

Especificaciones del programa principal

Solo el programa principal es capaz de invocar funciones y clases de python y java,

Tipos de datos

Los tipos de datos que se utilizarán son:

- int
- char
- float

NOTA: La variable puede o no ser iniciada al momento de definirla.

Arreglos

Arreglos de N dimensiones de cualquier tamaño. Los arreglos pueden ser de cualquier tipo.

```
Tipo        arreglo[dim1];
Tipo        arreglo[dim1][dim2][dim3];
```

NOTA: Al momento de declarar un arreglo, las dimensiones pueden ser expresadas usando literales enteras, constantes o expresiones aritméticas sobre literales y constantes.

Al momento de usar un arreglo, dentro de las dimensiones podrán venir expresiones aritméticas, arreglos, funciones, constantes, etc. Ejemplo:

```
int arreglo[25+4][CONSTANTE_ENTERA];
x = vector[5*4/7+8*9(4+2)][matriz[1][2*7]];
```

Constantes

Definición de constantes:

```
const tipo nombre_constante = valor;
```

```
const float pi = 3.14159;
const char c = 'X';       // X es una constante tipo char
const int X = 10;   // X es un tipo int
```

Operadores

Todos los operadores aritméticos, números enteros y reales ambos pueden ser negativos, los operadores aritméticos pueden ser: +, -, *, /, % (modulo ó residuo), = asignación y paréntesis.

Ejemplo

```
var = ( 1 + 2 – 3 * 4 / 5 ) % (5 * -3);
```

Comentarios

Los comentarios de línea o de bloque que se hagan deberán de ser pasados a código tres direcciones y al código ensamblador.

```
//Comentario de línea
/*Comentario
de bloque*/
```

Sentencia if-else

En la instrucción If, dentro de la condición pueden venir expresiones aritméticas y relacionales (<, >, = =, !=), esto implica operadores && (and), || (or), !(not); con o sin Else.

```
if ( condición )
{
    SENTENCIAS;
}
else
{
    SENTENCIAS;
}
```

Donde condición puede ser cualquier condición que involucre operadores aritméticos, relacionales y lógicos, y cualquier tipo de variable. Si el valor resultante de la condición es mayor que 0, la condición es VERDADERA, si el resultado de la condición es menor o igual a 0 la condición es FALSA. Ejemplo:

(VarAr[1] + 1 > var2 * 2 && var3 != var4)

Sentencia switch

La instrucción Switch, con n case y con o sin Default.

```
switch  (variable)
{
      case valor1:
                        SENTENCIAS;
                        break;
      case valor2:
                        SENTENCIAS;
                        break;
      ….                ….
      case valorn:
                        SENTENCIAS;
                        break;
      default:
                        SENTENCIAS;
                        break;
}
```

Ciclo for

for (variable = valor_inicial; condición; incremento|disminucion/accion posterior)

```
{
    SENTENCIAS;
}
```

Valor_inicial será un número entero o una variable tipo entero, y accionposterior es la accion que se ejecuta al finalizar una iteracion, condición puede ser cualquier condición, refiérase a la parte condición de la instrucción if.

La variable de asignación puede ser declarada antes o inicializada directamente en la misma asignación.

Ciclo while

```
while ( condición)
{
    SENTENCIAS;
}
```

Condición puede ser cualquier condición, refiérase a la parte condición de la instrucción if

Ciclo do while

```
do
{
    SENTENCIAS;
}
while ( condición );
```

Condición puede ser cualquier condición, refiérase a la parte condición de la instrucción if.

Instrucción continue y break

Los ciclos pueden incluir la palabra reservada continue cuya función es obviar las instrucciones siguientes e iniciar la siguiente iteración.

La palabra reservada break es opcional e indica que se finaliza cualquier ciclo o sentencia de control sin ejecutar el código que se encuentre por debajo de esta palabra.

Las instrucciones continue y break dependen del contexto por lo que están sujetas a errores semánticos.

Llamadas a funciones y procedimientos

Si en dado caso la funcion o el procedimiento llevaran parametros, estos pueden ser:

Por valor: tipo_var1 nombre_var1, tipo_var2 nombre_var2, ... , tipo_varn nombre_varn

Por referencia: tipo_var1 &nombre_var1, tipo_var2 &nombre_var2, ... , tipo_varn &nombre_varn

Invocación (activación):

Variable = PY.nombre_funcion(parámetro);

PY.nombre_procedimiento(parámetros);

Si la firma de una funcion/procedimiento invocado coincide con varias funciones/procedimientos, entonces se activa la que se encuentra mas abajo en la seccion de declaraciones de librerias.

scanf

Esta instrucción permitirá asignar valores a las variables.

scanf(“ mascara “, &variable);

Mascara se refiere a texto y al indicador de que tipo de dato leerá, ejemplo:

scanf(“Valor %d”, &var); //Leerá del teclado un valor para asignar a variable tipo int

scanf(“Valor %c”, &var); //Leerá del teclado un valor para asignar a variable tipo char

scanf(“Valor %f”, &var); //Leerá del teclado un valor para asignar a variable tipo float

printf

Esta instrucción permitirá desplegar mensajes y los valores de las variables

printf(“ texto “); ó printf(“El valor es mascara “, var);

En la primera instrucción solo se mostrará texto, en la segunda se mostrará el texto y el valor que contenga la variable var, puede haber más de una variable por mostrar.

Mascara se refiere a texto y al indicador de que tipo de dato desplegará, ejemplo:

printf(“Valor %d”, var); //Desplegará el valor de una variable tipo int

printf (“Valor %c”, var); //Desplegará el valor de una variable tipo char

printf (“Valor %f”, var); //Desplegará el valor de una variable tipo float

clrscr

limpiar la pantalla, ejemplo:

clrscr();

getch

Esta instrucción leerá una tecla del teclado para poder seguir la ejecución del programa. El valor que regresa puede o no ser asignado a una variable tipo char o int, si es asignado a una

variable tipo char la variable tendrá el carácter que se presionó; si es asignado a una variable tipo int la variable tendrá el valor ASCII del carácter que se presionó.

Clases

Todas las clases serán públicas, por lo que deben incluir el modificador de visibilidad ‘public’, sin embargo, los miembros pueden incluir alguno de los modificadores ‘public’ y ‘private’.

Las clases deben soportar herencia simple.

El uso de las clases se maneja de la siguiente manera

Declaración

```
JAVA.Nombre_Clase    Nombre_Var1, Nombre_Var2;
JAVA.Nombre_Clase    Nombre_Var1(parámetros); // llamando a constructor
```

Llamada a métodos

```
Variable = JAVA.nombre_objeto.método(parámetros);
JAVA.nombre_objeto.método(parámetros);
```

Si el nombre de una clase coincide con varias clases, entonces se usa la que se encuentra mas abajo en la sección de declaraciones de librerias