Diseño de Compiladores

Manual de Usuario

Mayka

Guillermo Enrique Valles Villegas – A01561722

25 Noviembre 2020

Índice

¡Bienvenido!

Muchas gracias por decidir utilizar **Mayka** como tu próximo lenguaje de programación.

Mayka es un lenguaje estilo C que te permite hacer programas y visualizar resultados con un output grafico si así lo deseas, aunque no es obligatorio hacerlo.

Tu rango de operaciones aritméticas con Mayka es bastante amplio. Puedes hacer todas las operaciones básicas y lógicas. Mayka también puede leer paréntesis por si deseas generar resultados con orden diferente del jerárquico usual.

A continuación, podrás encontrar la estructura y descripción que tu código debe seguir para que **Mayka** lo pueda ejecutar correctamente.

¡Comencemos!

Lo Principal.

Antes de continuar, deber saber las reglas principales de Mayka:

1. Al final de cada línea de código, debes incluir un punto y coma “;”.
2. Mientras que no es obligatorio declarar variables o funciones, debes poner el nombre del programa y un main de la siguiente manera.

```
program nombre_Programa;

main ( ) {

}
```

Variables.

Con Mayka, no debes preocuparte por estar buscando donde declaraste una variable, ya que la declaración de variables solo se permite luego de tu nombre del programa y antes de las funciones.

La sintaxis de declaración es la siguiente:

```
var tipo : variable;
```

O bien, puedes usar la siguiente manera para declaración de variables del mismo tipo:

```
var tipo : variable1, variable2,...variableN;
```

Mayka solo soporta los tipos Enteros, Flotantes y Caracteres para sus variables.

```
program calcular;
var int : x, y;
var float : a;
main ( ) {

}
```

Variables con Dimensiones.

Además de las variables simples, con Mayka puedes crear arreglos y matrices.
Para esto, solo debes cambiar un poco como declaras tu variable, agregándole las dimensiones de esta manera:

```
var tipo : variable[entero];
```

O bien, puedes usar la siguiente para matrices:

```
var tipo : variable[entero1, entero2];
```

Para llamar la variable en la dirección que deseas, se hace de la siguiente manera:

```
x = a[1];
y = b[0,3];
```

Funciones.

Mayka te permite facilitar tu código con funciones, las cuales deben ser declaras antes de tu main. Al igual que las variables, las funciones deben tener un tipo, solo que además de Enteros, Flotantes y Caracteres, las funciones pueden ser de tipo Void si no quieres que regrese valor.

La sintaxis de declaración de funciones es la siguiente:

```
program calcular;

tipo module nombre_Funcion ( parametros );
var tipo : variable1, variable2,...variableN;
{
        estatutos;
}
```

Los parámetros y variables dentro del modulo son completamente opcionales. Para los parámetros, debes seguir el siguiente formato:

```
tipo variable1, tipo variable2...tipoN, variableN
```

Además de los estatutos, las funciones que no sean de tipo Void, deben tener un ***return***, que es lo que devolverá al asignar la función a una variable.

```
return (variable);
```

Las llamadas a funciones dentro del main pueden ser de dos maneras:

```
variable  = funcion(parametros);
```

De esta manera si es de return, o bien, de la siguiente manera si es Void:

```
call funcion(parametros);
```

Ciclos.

Si quieres repetir alguna operación cierto número de veces, Mayka te ofrece dos posibilidades, los ciclos **For** y los ciclos **While**.

Ambos te pueden ayudar de igual manera, sin embargo, hay ciertas diferencias en ambos que debes saber además de su sintaxis.

Los ciclos For aumentan de uno en uno el valor de su variable automáticamente, mientras que los ciclos While no lo hacen de manera automática, ¡por lo que se pueden quedar ciclados si no cambias el valor de la variable!

Para **For**, su sintaxis es la siguiente:

```
for  variable = expresion to expression do {
                    estatutos;
}
```

Para **While**, la sintaxis es:

```
while (  expresion ) do {
          estatutos;
}
```

Decisiones.

Para tomar decisiones dentro del código, Mayka utiliza los IF, con la siguiente estructura:

```
if ( expresion ) then {
        estatutos;
} else {
        estatutos;

}
```

Si el resultado de la expresión es verdadero, ejecuta el primer estatuto, de lo contrario, va al estatuto dentro del *else*.

Entrada y Salida.

Con tu terminal, Mayka puede mostrarte tus resultados, o bien, recibir datos. Para esto, se tienen las funciones **Read** y **Write**.

Read te permite leer el valor de una variable (previamente declarada) desde consola.
La sintaxis es la siguiente:

```
read (variable);
```

Write imprime el valor de una variable o algún letrero en consola.
Su sintaxis es la siguiente:

```
write (variable);
write (letrero);
write (variable, letrero...);
```

Un letrero solo puede ser UNA palabra. O sea, solo puede contener letras.

Salida Grafica.

Si deseas visualizar tus resultados fuera de la terminal, o simplemente quieres pasar el tiempo creando formas interesantes, Mayka te permite ciertas funciones especiales para esto.

A continuación, podrás encontrar cada una con su sintaxis, descripción y un ejemplo.

Line

La función Line toma dos parámetros, que funcionan como coordenadas X y Y en un plano cartesiano.

```
line (expresion, expresion);
```

Point

La función es bastante simple. No necesita ningún parámetro ya que solo genera un punto donde el puntero este colocado.

```
point ( );
```

Circle

Es una función que toma como parámetro un solo dato, que puede ser flotante o entero. Este dato funciona como el radio, que Mayka utilizará para generar un circulo.

```
circle (expresion);
```

Arc

Al igual que Circle, Arc toma un solo parámetro, solo que Arc dibujará solamente la mitad del circulo.

```
arc (expresion);
```

Penup

¿Quieres mover tu puntero sin que se arruine tu canvas? ¡No te preocupes! Con Penup puedes hacerlo.
Penup no necesita ningun parámetro, ya que solamente le dice a Mayka que deje de dibujar, así puede mover tu puntero con otras funciones.

```
penup ( );
```

Pendown

Para poder seguir dibujando luego de utilizar Penup, debes usar Pendown. Posteriormente, el puntero seguirá dibujando como usual.

```
pendown ( );
```

Color

Si quieres llenar tus graficas con colores puedes hacerlo con la función Color.
Color toma tres parámetros que se conocen como RGB (Red, Green y Blue). Estos valores pueden ser desde 0 hasta 255 cada uno.

```
color (expresion, expresion, expresion);
```

Size

Con la función Size, puedes cambiar el tamaño del puntero para que los trazos se vean diferentes, solo debes pasarle el parámetro del tamaño que deseas.

```
size (expresion);
```

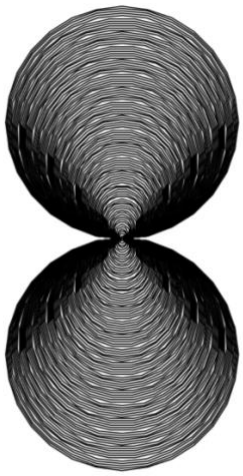
Clear

La ultima función grafica con Mayka. Clear te permite borrar el canvas sin tener que parar y volver a correr tu código, así puedes visualizar varios resultados diferentes en una sola pasada de código.

```
clear ( );
```

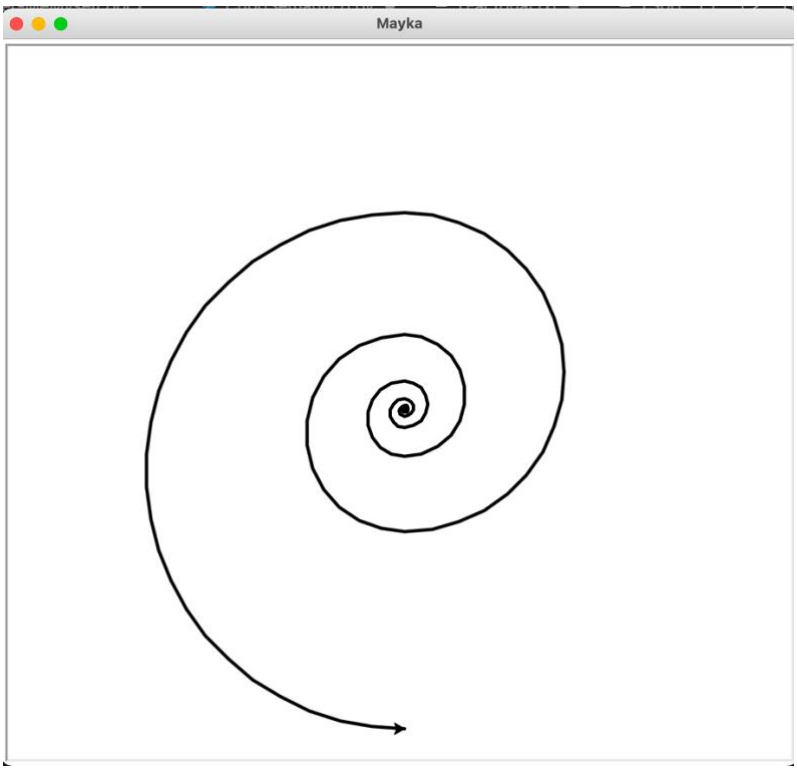
Algunos Ejemplos.

¡Esta es una figura que puedes hacer muy fácilmente con Mayka! ¿Qué necesitas? Solamente dos ciclos anidados.



```
i = 0;
    j = 0;
    while(i <= 100) do{
        while(j <= 100)do{
            write(i,j);
            circle(i);
            j = j*-1;
            circle(j);
            j = j*-1;
            i = i+1;
            j = j+1;
        }
    }
```

¿O qué tal la serie de Fibonacci?




```
program fibonacci;
var int : n,
first,second,next,c,s;
main(){
    first = 0;
    second = 1;
    n = 14;
    for c = 0 to n do {
        if(c<=1) then{
            next = c;
        }else{
            next = first + second;
            first = second;
            second = next;
        }
        size(3);
        write(next);
        arc(next);
    }
}
```

¡Unos círculos de colores diferentes!

```
program sphere;
var int : a, b, c;


main(){
    size(10);
    penup();
    line(30,30);
    color(255,0,0);
    pendown();
    circle(20);

    penup();
    line(0,-10);
    color(0,255,0);
    pendown();
    circle(20);

    penup();
    line(10,0);
    color(0,0,255);
    pendown();
    circle(20);

}
```

Así como estos ejemplos hay muchas posibilidades por hacer.

Muchas gracias por elegir Mayka.

¿No eres muy fan de la documentación? No te preocupes. Puedes revisar este link para encontrar un video demostrativo con información extra y ejemplos.

https://drive.google.com/file/d/1TpSRff-TOeqhA_2SKC1RBe05Ecm7UXZv/view?usp=sharing