

## Lenguaje PsiCoder

### Manual de Referencia

PsiCoder es una herramienta diseñada para facilitar el proceso de enseñanza y aprendizaje de los conceptos básicos en la introducción a la programación. Cuenta con los comandos básicos encontrados en lenguajes como java, C++, python, entre otros. A continuación se muestra de manera detallada los comandos usados en el lenguaje PsiCoder:

1. Todo programa en PsiCoder debe contener para su funcionamiento la siguiente estructura básica:

```
//declaración de variables globales, funciones y estructuras

funcion_principal

//código que se ejecutará en el lenguaje

fin_principal
```

2. **Declaración de variables:** Cuenta con 5 tipos de datos primitivos:

- ❑ **booleano:** variable que toma los valores de ‘verdadero’ o ‘falso’
- ❑ **caracter:** variable que toma el valor de cualquier carácter [a – zA – Z0 – 9 ]. Sin incluir la ‘ñ’, ni ‘Ñ’.
- ❑ **entero:** variable que toma valores enteros positivos o negativos en el rango [-2147483648,2147483647]
- ❑ **real:** variable que toma valores reales positivos o negativos en el rango [-2147483648,2147483647]
- ❑ **cadena:** variable compuesta por 0 o más caracteres y espacios en blanco

La asignación se hace de igual forma que en varios lenguajes conocidos, anteponiendo el tipo de dato al nombre de la variable. Veamos un ejemplo:

```
funcion_principal
booleano flag = verdadero;
caracter letter = 'x';
entero count = 7;
real mean = 3.5;
cadena name = "Lenguajes de programacion";
entero x,y=1,z=0;
fin_principal
```

Tenga en cuenta lo siguiente:

- cuando se hace uso de un carácter se deben utilizar comillas simples.
- Si se trata de una cadena de 0 o más caracteres se debe utilizar siempre las comillas dobles
- En los valores reales, siempre se usa el punto decimal(.)
- Si se desea crear más de una variable en la misma línea, estas van separadas con coma(,) y cada una puede tomar o no un valor inicial

### 3. Lectura de datos

La lectura de datos por consola se hace como sigue:

```
funcion_principal  
    entero a;  
    leer( a );  
fin_principal
```

- Solo es permitido leer un valor con el comando `leer`.
- La variable que se desea leer siempre debe estar encerrada entre paréntesis '()''

### 4. Imprimir:

La impresión por consola permite mostrar cualquier variable, operación, ecuación, método o texto.

```
funcion_principal  
    caracter letter = 'x';  
    entero cont = 7;  
  
    imprimir( "letra" , letter , " contador " , (cont * cont) );  
    imprimir( "total" , cont * 2 + 1 );  
fin_principal
```

- La forma en que se concatenan una o más variables y/o cadenas es por medio de la coma(,).
- Dentro de cualquier expresión, como en este caso en el comando escribir, se pueden realizar operaciones matemáticas (con o sin paréntesis): +, -, \*, /, %.

## 5. Condicionales:

- Estructura si entonces:

```
funcion_principal
    entero a= 3;
    si ( a < 5 && a>0 ) entonces
        imprimir(a);
    fin_si
fin_principal
```

Siempre que se crea un condicional si, este debe terminar con la palabra ‘fin\_si’. Además, en los condicionales y las estructuras de control cíclicas (mientras, para, hacer .. mientras) la condición siempre debe estar encerrada entre paréntesis ‘( )’.

- Estructura si-no:

```
funcion_principal
    entero a= 3;
    si ( (a < 5) && (a>10) ) entonces
        imprimir(a);
    si_no
        imprimir(a-1);
    fin_si
fin_principal
```

## 6. Ciclos

- Estructura Mientras

```
funcion_principal
    entero a = 3;
    mientras ( a > 0 ) hacer
        a = a - 1;
    fin_mientras
fin_principal
```

Siempre que se crea un ciclo mientras, este debe terminar con la palabra ‘fin\_mientras’.

- Para

La estructura del ciclo para se divide en 3 partes:

➤ variable del ciclo

- condición de terminación
- variable o número que indica el paso de la variable en el ciclo

```
funcion_principal
    entero a = 3;
    entero p = 2;
    entero j = 2;
    para ( entero i = 0 ; i < a ; 1 ) hacer
        imprimir(i);
    fin_para

    para ( j = 0 ; i < a || j < 100 ; p ) hacer
        imprimir(j);
    fin_para
fin_principal
```

- Hacer-Mientras

```
funcion_principal
    entero a = 3;

    hacer
        a = a/2;
    mientras( a > 0 );
fin_principal
```

## 7. Selección múltiple

```
funcion_principal
    entero a = 3;
    leer(a);
    seleccionar ( a ) entre
        caso 0 :
            imprimir(a);
            romper;
        caso 1:
            imprimir(a*2);
            romper;
        defecto:
    fin_seleccionar
fin_principal
```

## 8. Estructura

Una estructura nos permite agrupar varios datos ya sean primitivos o estructuras.

```
estructura C
    entero a;
fin_structura

estructura Point
    entero x;
    entero y;
    real a,b,c;
    C c;
fin_structura

funcion_principal

    Point p;
    p.x = 5;
    p.y = 10;
    p.c.a = 5;

fin_principal
```

- Para hacer uso de una estructura debe ser creada antes
- No existe un límite en el número de variables que contiene una estructura.
- Los valores de la estructura son accedidos por medio de un punto(.)

## 9. Función

Toda función contiene la siguiente estructura:

```
funcion entero sum(entero a, entero b) hacer
    retornar a + b;
fin_funcion

funcion_principal

    imprimir (sum(3,5))

fin_principal
```

- La función puede tener 0 o más parámetros
- Siempre debe retornar un valor correspondiente al tipo de la función

- Para poder llamar una función dentro de otra o en la función principal esta se debe crear antes.

## 10. Comentarios

Los comentarios tienen la siguiente estructura

```
funcion_principal

// imprimir (sum(3,5))
/*
    comentario de
    varias líneas

 */

fin_principal
```

Ejemplo programa completo:

```
funcion_principal
    entero n;
    booleano flag = falso;
    leer( n );
    para (entero i = 2 ; i < n ; 1 ) hacer
        si ( n % i == 0 ) entonces
            imprimir( "no es primo");
            flag = verdadero;
        fin_si
    fin_para
    si ( flag == falso ) entonces
        imprimir("es primo");
    si_no
    fin_si
fin_principal
```