**Universidad Nacional Autónoma de México**

**Facultad de Ciencias – Ciencias de la Computación**

**Sistemas Operativos. Semestre 2019-2**

**Estrada Gómez César Derian. 31222446-4**

## Tarea 1. Introducción a C.

### 1. *¿Qué diferencias hay entre Java y C?*

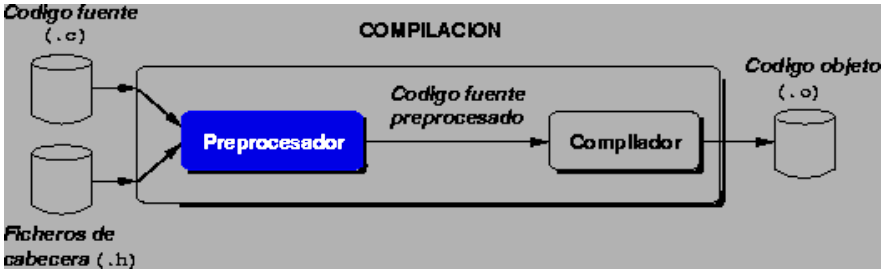
| C | Java |
|---|---|
| Programación estructurada. | Programación Orientada a Objetos. |
| Lenguaje rápido y compilado. | Java necesita una compilación previa para poder ser ejecutado. |
| A menos que sea escrito en ANSI C, C no es portable. | Es extremadamente portable; puede correr en cualquier arquitectura. |
| Lenguaje de bajo nivel con instrucciones de alto nivel.<br>*Menos fácil, más control.* | Lenguaje con instrucciones de bajo nivel, con poco acceso al sistema y menor gestión de memoria.<br>*Más fácil, menos control.* |
| Se pueden escribir virus en C debido a su libre acceso a memoria. | Es seguro. No se escriben virus escritos en Java. |
| **Fichero: holamundo.c**<br><br>`#include <stdio.h>`<br>`int main(){`<br>`  printf("Hola mundo");`<br>`  return 0;`<br>`}` | **Fichero: holamundo.java**<br><br>`public class holamundo{`<br>`  public static void main(String[] args){`<br>`  System.out.println("Hola mundo");`<br>`  }`<br>`}` |

### 2. *¿Qué es el Preprocesador y qué salida genera?*

En el proceso de compilación de programas en C, la primera etapa de este proceso se le conoce como *Preprocesamiento*. El preprocesador de C es una herramienta que *filtra* el código fuente antes de ser compilado. El preprocesador acepta como entrada código fuente y se encarga de:

- Eliminar los comentarios
- Interpretar y procesar las directivas de procesamiento.

Genera un código preprocesado que es usado por el compilador para generar el código objeto.



***3. Para cada secuencia de instrucciones, menciona si es correcto o no, si no lo es justifica tu respuesta:***

***a) Es correcto.***

```
1 int main(){
2    long int a = 34;
3    char b = 'A';
4    long c = a * b;
5    return 0;
6 }
```

***b) No es correcto.***

```
1 #define VARIABLE 777
2 int main(){
3    if(VARIABLE){
4        VARIABLE = 1;
5    }
3    for(;;){
4        return 0;
5    }
6 }
```

No es correcto debido a que provoca un error *lvalue*.

*lvalue* significa valor del lado izquierdo, particularmente, es el valor del lado izquierdo de un operador de asignación.

Concretamente, en este código, este error aparece en la línea 4, donde a la constante 'VARIABLE' previamente definida le queremos asignar un nuevo valor, en este caso de 1.

Cuando se ha definido una constante no se le puede reasignar un nuevo valor.

***c) Es correcto.***

```
1 #include "stdio.h"
2 #define EQ(x,y) (x==y)
3 int main(){
4    unsigned int a = -1;
5    unsigned int b = a << 1;
6    if(eq(a,b) == EQ(a,b)){
7        printf("Pase con %d y %d",a,b);
8    }
9    return 0;
10 }
11 int eq(int a, int b){
12    return a == b;
12 }
```

***4. ¿Pasaría por error sin el preprocesador y compilador?***

```
1 #ifdef _MIMACRO_
2 #define PI 3.1416
3 #endif
4 #define CONFIRMA(X) (X > 40.0)
4 #define MIMACRO
6 int reconfirma(float n);
5 int main(){
6    //radio del circulo
7    float radio = 3.4;
9    //area del circulo
8    float area = PI * radio * radio;
10   while(1){
11       /* si el area es menor que 40
12       va aumentando el area,
13       sino rompe el ciclo y termina */
14       if(area < 40.0){
15          area += 4.4;
16          continue;
17       }
18       break;
19    }
20    int resultado = reconfirma(area);
21    return 0;
22 }
23 //funcion para reconfirmar
23 int reconfirma(float n){
24      return CONFIRMA(n);
25 }
```

*Por el preprocesador pasaría sin ningún error debido a que las directivas están bien declaradas.*

*Sin embargo, al momento de pasar por el compilador, marcaría error con la constante PI porque está fue declarada dentro de un marco que en ningún momento se ejecuta dentro del código.*