Introducción a la programación en C

Operadores y funciones

Bach. Constantino Bolaños Araya

Escuela de Veranillo en HPC, 2018

Operaciones y funciones

- Los datos deben ser transformados con operaciones o funciones.
  - *Operadores:* conjunto básico de transformaciones.
    - Cada lenguaje de programación define sus propios operadores.
    - Algunos operadores comunes: `() [] -> . ! ~ ++ -- + - * / % ^ < <= > >= == != & | && || ?: = += -= *= /= %= ^= |= << >> <<= >>=`
  - *Funciones:* subrutinas, métodos, procedimientos, etc. Similares a los operadores pero con diferencias sintácticas.
    - En su mayoría definidas por el usuario.
    - En su prototipo o especificación se detallan sus argumentos y valor de retorno
    - Ejemplo: `int sumaDeEnteros(int a, int b)`

Operadores en C

Operadores aritméticos

- +: suma
- −: resta
- ∗: multiplicación
- /: división
- %: módulo (residuo)
- ++: incremento
- −−: decremento

Operadores relacionales

- ==: “igual a”
- !=: “diferente de”
- >: “mayor que”
- <: “menor que”
- >=: “mayor o igual que”
- <=: “menor o igual que”

Operadores lógicos

- `&&`: Y lógico (1 && 1 es verdadero, 1 && 0 es falso)
- `||`: O lógico (1 || 0 es verdadero, 0 || 0 es falso)
- `!`: NO lógico (!0 es verdadero, !1 es falso)

Operadores de asignación

- =: asignación
- +=: sumar y asignar
- −=: restar y asignar
- *=: multiplicar y asignar
- /=: dividir y asignar
- %=: módulo y asignar

Operadores misceláneos

- `sizeof()`: tamaño de una variable
- `&`: dirección en memoria de una variable
- `*`: puntero a una variable
- `? :`: expresión condicional

Precedencia de operadores

- La precedencia de operadores está establecida en C (ver este sitio para referencia).

```
float a = (10 * 6 + 25 / 5) + 9 % 3 + 10;
```

Funciones en C

Definiendo una función

- Una función tiene cuatro componentes esenciales:
  1. Tipo de retorno
  2. Nombre
  3. Parámetros
  4. Cuerpo

```
float celsiusAFahrenheit(float grados_C) {
    return grados_C * 1.8 + 32;
}
```

Llamando a una función

- Para llamar a una función, basta con invocarla por su nombre, y proveyendo los argumentos apropiados.
- Cuando hay un llamado a función, el flujo del programa "salta" al cuerpo de la función; y retorna de donde se llamó originalmente cuando la función termina.

```
int main() {
    float tempCelsius = 23.4;

    float tempEnFahrenheit = celsiusAFahrenheit(tempCelsius

    printf("La temperatura en F: %f", tempEnFahrenheit);

    return 0;
}
```