Modelo de Respuestas Esperadas

IMPERATIVE PROGRAMMING

Imperative programming, as opposed to declarative programming, is a programming paradigm **that** describes computation in terms of a program state and statements **that** change the program state. In such the same way as the imperative mood in natural languages expresses commands to take action, imperative programs are a sequence of commands for the computer to perform.

Imperative programming languages stand in contrast to other types of languages, such as functional and logical programming languages. Functional programming languages, such as Haskell, are not a sequence of statements and have no global state as imperative languages do. Logical programming languages, like Prolog, are often thought of as defining "what" is to be computed, rather than "how" the computation is to take place, as an imperative programming language does.

The hardware implementation of almost all computers is imperative; nearly all computer hardware is designed to execute machine code, **which** is native to the computer, written in the imperative style. From this low-level perspective, the program state is defined by the contents of memory, and the statements are instructions in the native machine language of the computer. Higher-level imperative languages use variables and more complex statements, but **they** still follow the same paradigm. Recipes and process checklists, while not computer programs, are also familiar concepts **that** are similar in style to imperative programming; each step is an instruction.

A. What do these words refer to in the text?

1. **that** (line 1) programming paradigm
2. **that** (line 2) statements
3. **which** (line 12) machine code
4. **they** (line 15) higher-level imperative languages
5. **that** (line 17) familiar concepts

B. Answer the following questions in Spanish.

1. What is imperative programming?

Es un paradigma de programación que describe la computación con respecto al estado del programa y las sentencias que cambian el estado del programa.

2. Why does an imperative language act in the same way as the imperative mood in the natural language?

Porque expresan comandos a ejecutar. / En el caso de los lenguajes imperativos la secuencia de comandos es ejecutado por la computadora.

3. How do imperative languages differ from functional and logical languages?
Porque los lenguajes funcionales no son una secuencia de sentencias y no tienen estado global como los imperativos y los lógicos definen lo que se va a computar a diferencia de los imperativos que determinan de qué manera (cómo) se va a hacer la computación (cálculo).

4. What is imperative style used for?
Es usado para la implementación del hardware ya que la mayoría de los hardware están diseñados para ejecutar código de máquina escrito en un estilo imperativo. / Para describir códigos de máquina.

C. Translate the following text into Spanish.

Assignment statements, in general, perform an operation on information located in memory and store the results in memory for later use. High-level imperative languages, in addition, permit the evaluation of complex expressions, which may consist of a combination of arithmetic operations and function evaluations, and the assignment of the resulting value to memory. Looping statements allow a sequence of statements to be executed multiple times. Loops can either execute the statements they contain a predefined number of times, or they can execute them repeatedly until some condition changes. Conditional branching statements allow a block of statements to be executed only if some condition is met. Otherwise, the statements are skipped and the execution sequence continues from the statement following the block. Unconditional branching statements allow the execution sequence to be transferred to some other part of the program. These include the jump, called "goto" in many languages, and the subprogram or procedure call.

Las sentencias de asignación, en general, ejecutan una operación sobre la información ubicada en la memoria y almacenan los resultados en la memoria para ser usados posteriormente. Los lenguajes imperativos de alto nivel, además, permiten la evaluación de expresiones complejas, que pueden consistir en una combinación de operaciones aritméticas y evaluaciones de funciones, y la asignación del valor resultante a la memoria. Las sentencias en bucle (en ciclo) permiten que una secuencia de sentencias se ejecute varias veces. Los bucles pueden ejecutar las sentencias que contienen un número predeterminado de veces, o pueden ejecutarlas repetidamente hasta que cambien algunas condiciones. La sentencias de bifurcación condicional permiten que se ejecute un bloque de sentencias sólo si cumple con alguna condición. Si no, las sentencias se saltean y la secuencia de ejecución continúa desde la sentencia que sigue al bloque / la sentencia a continuación del bloque. Las sentencias incondicionales de bifurcación permiten que la secuencia de ejecución se transfiera a alguna otra parte del programa. Estas incluyen al salto, llamado “goto” (ir a: instrucción simbólica de salto incondicional) en muchos lenguajes, y al subprograma o llamada a un procedimiento.