**FACULTAD DE INFORMÁTICA**

**NOMBRE Y APELLIDO: ........................................................... AYUDANTE: ................................**

**LEGAJO Nº........................ e-mail: ………………………….. FECHA: ........................................**

***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.

1. **What do these words refer to in the text?**

1. **that**  (line 1) Imperative programming

2. **that**  (line 2) statements

3. **which** (line 12) machine code

4. **they**  (line 15) Higher-level impertative languages

5. **that**  (line 17) \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Answer the following questions in Spanish.**

1. What is imperative programming?

Es un paradigma de programación que describe la computación en términos de estado de un programa o sentencias que cambian el estado de un programa.

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

Porque en la misma forma que el lenguaje natura expresa los comandos para realizar una acción, la programación imperativa es una secuencia de comandos que debe ejecutar la computadora

3. How do imperative languages differ from functional and logical languages?

El lenguaje funcional no tiene una secuencia de instrucciones y tampoco un estado global como el imperativo, el lenguaje locgico a diferencia del imperativo no define como la computadora resuelve el problema.

4. What is imperative style used for?

.............................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

1. **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, a predefined number of times, the statements they contain, 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 sentencia de asignación, en general, ejecutan una operación en información localizada en memoria y guardan los resultados en memoria después usarlos. Los lenguajes imperativos de alto nivel, además permiten la evaluación de expresiones complejas, las cuales pueden consistir en una combinación de operaciones aritméticas y evaluación de funciones, y la asignación del valor resultante en memoria. Las sentencias de bucle permiten que una secuencia de sentencias se ejecute múltiples veces. Los bucles pueden ejecutar las sentencias que tienen un numero predefinido de veces o pueden ejecutarla repetidamente hata que alguna condición cambie. Las sentencias de salto condicional permiten que un bloque de sentencias sea ejecutado solo si alguna condición se cumple. De otra forma, las sentencias son salteadas y la ejecución contunua por el siguiente bloque de sentencias. Las sentencias de salto incodicional permiten que la secuencia de ejecución sea transferida a otra parte del programa. Esto incluye el salto llamado “goto” en algunos lenguajes y el subprograma o llamada al procedimiento

...................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

.......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

.......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

.......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

.......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................