(1) ¿Qué diferencia la programación estructurada de los lenguajes de alto nivel?

Según el texto, la principal diferencia es que, mientras que en un lenguaje de alto nivel el conjunto de abstracciones están predefinidas por el diseñador del lenguaje, y el programador no debería preocuparse por su implementación sino por sus parámetros y salidas, en un lenguaje estructurado es el programador quien se ocupa de definir las abstracciones que le van a ser útiles

(2) ¿Cómo se resuelve un problema con el paradigma de programación estructurada?

Mediante un proceso de "descomposición sucesiva", el cual consiste en resolver el problema mediante las operaciones y objetos necesarios, sin importar que estén definidos o no. Se preocupa de las reglas que deberían cumplir estas herramientas u objetos, y, mediante ellas, resuelve el problema en una máquina abstracta (En su mente).

Sigue el mismo procedimiento para concidierar la creación de cada uno de estos objetos (o herramientas) como un problema y descomponerlo en otros objetos de un menor nivel de abstracción.

Así, siempre que el nivel de abstracción inicial no sea "infinito", acabaremos llegando a unos tipos de datos primitivos (descritos en el lenguaje nuclear), y podremos construir las herramientas dentro del lenguaje, desde abajo hacia arriba.

¿Qué se entiende por máquina abstracta?

Es la lógica determinada por los axiomas que definen a los objetos (o herramientas) construidos en cada una de las partes descritas en el primer apartado de la pregunta.

En un inicio esta máquina debe ser "ejecutada" por el programador en su cabeza, pero el objetivo es acabar construyendo estructuras y algoritmos en el lenguaje que nos permitan aplicar esta máquina abstracta en el ordenador.

(3) Describe ejemplos de las distintas abstracciones que dan los autores

- Tipos de datos primitivos: predefinidos por el lenguaje de programación y tratados por el compilador (int o Float en C)
- Procedimientos y Funciones: Agrupación de código que resuelve una tarea específica, ocultando la complejidad de la implementación subyacente.
- Módulos: Agrupación de código que permite organizarlo en secciones más pequeñas, sin llegar a ocultar el procedimiento

(4) Según los autores, ¿qué describe un tipo abstracto de datos?

Define una clase de objetos, o herramientas abstractas, las cuales a su vez están definidas por las operaciones realizables con ellas.

(5) Según los autores, ¿qué distingue un TAD de un tipo predefinido?

Que, en el caso de un tipo predefinido, la abstracción se realiza dándonos muy pocos detalles; se ocupan el lenguaje y compilador de interpretar los bits y realizar operaciones con ellos.

Sin embargo, un TAD se construye dentro del lenguaje, definiendo las operaciones que puede realizar, de los cuales podemos llegar a tener más información; esto se implementa en lo conocido como "operation cluster".

(6)

Un operation cluster es un tipo de herramienta abstracta con varias funciones que se pueden utilizar para resolver procedimientos complejos sin necesidad de preocuparse de todos los pasos que hay detrás, y está definida precisamente por estos procedimientos.

Los equivalentes son las clases en POO, las bibliotecas de C, o las interfaces de Java.

(7) "is"; por ejemplo:

```
class : cluster(element-type: type)
        is [list of operations]
```

(8) "rep"; por ejemplo: rep(type-param: type) = ([name-atributte: type])

(9) "create"; por ejemplo:

```
create
    [operación que crea valores]
    return valores
end
```

(10) Mediante el uso de variables, las cuales, en caso de estar declaradas para representar objetos de un tipo, pueden comportarse como tal.

En C++ es equivalente a la creación de variables que sean objetos instanciados de una clase, o a la reserva de un espacio en memoria que pueda almacenar instancias de la clase.