*Resumen Final CADP*

Un **programa** es un conjunto de órdenes o instrucciones ejecutables sobre una computadora que permiten cumplir con una función específica (dichas instrucciones están escritas en un lenguaje de programación concreto).

**Modularizar:** Significa dividir un problema en partes funcionalmente independientes, que encapsulan operaciones y datos.

Se trata de separar tareas con datos propios y de comunicación perfectamente especificados que se comunican entre sí.

Un **dato** es una representación de un objeto del mundo real. Permiten modelizar los aspectos del problema que se quiere resolver mediante un programa ejecutable en una computadora. Hay datos constantes y datos variables. Cada dato debe tener asociado un *tipo de dato.*

Los **tipos de datos** son una forma de identificar y clasificar los datos utilizados en programación.

Cada uno tiene un rango de valores posibles, una representación interna (es decir, un número máximo y un número mínimo) y están ligados a un conjunto de operaciones permitidas específicamente para ese tipo de dato.

Resultan útiles para la gestión de la información.

Los tipos de datos pueden ser **simples o compuestos**.

Simples: puede tomar un único valor a la vez.

Compuestos: pueden tener varios valores a la vez (guardan alguna relación lógica entre ellos)

Una **constante** representa un dato que no cambia durante la ejecución de un programa. Este tipo de dato puede ser numérico, lógico o carácter.

Un **dato ordinal** es aquel que tiene sus elementos ordenadores discretamente.

Esto significa que para cada elemento (que es parte del tipo) existe un elemento anterior y otro posterior.

Por esta razón los tipos de datos enteros, carácter y lógico se denominan tipos de datos ordinales. Los reales quedan excluidos porque se sabe que entre dos

Un **tipo de dato definido por el usuario** es aquel que no existe en la definición del lenguaje, donde el usuario es el encargado de determinar su denominación y el conjunto de valores y operaciones que estarán permitidas para el mismo.

Los tipos de datos definidos por el usuario permiten un lenguaje con:

* Mejores posibilidades de abstracción de datos. Esto permite mayor claridad para la lectura de los programas.
* Mayor seguridad respecto de las operaciones que se llevan a cabo sobre cada clase de datos, permitiendo mayor número de validaciones.
* Límites preestablecidos sobre los valores posibles que pueden tomar las variables que corresponden al elemento del mundo real asociado.

Cada **declaración de tipos** (type) define identificadores que pueden ser utilizados en declaraciones de otros tipos, variables, procedimientos y funciones posteriores. Se deben utilizar nombres descriptivos para identificar los datos de un programa de forma tal que se pueda conocer su dirección real en la memoria y el valor que contiene.

Las ventajas que presentan son:

* Flexibilidad: en caso de que sea necesario modificar la forma en que se representa la información solo se debe modificar una declaración en lugar de una serie de declaraciones de variables.
* Documentación: se pueden asignar nombres de tipo identificadores que representan la manera en que deben ser usados y no como están constituidos, facilitando el entendimiento y la lectura del programa.
* Seguridad: con su declaración se reducen los errores de correspondencia entre el valor que se pretende asignar a una variable y el previamente declarado. Al realizar los chequeos sobre el código será más fácil distinguir errores, de esta manera pueden obtenerse programas más confiables.

Los tipos de datos definidos por el usuario nos permiten crear un nuevo tipo que se corresponda con un grupo de objetos del mundo real.

Este nuevo **tipo de dato** será **enumerativo**.

Un **procedimiento** es un conjunto de instrucciones que realizan una tarea específica y como resultado del cual puede retornar cero, uno o más valores como respuesta.

Una **función** es un módulo que realiza una tarea específica y que como resultado de ella retorna un único valor como respuesta.

Si bien tienen una función similar, existen **diferencias** entre estos dos módulos: Las funciones retornan siempre un único resultado, mientras que los procedimientos pueden retornar 0,1 o más valores, y las funciones indican en su definición el tipo de dato que retorna como resultado.

Se denomina **parámetros** a la serie de datos con los que se comunican los módulos. Los parámetros de un módulo deben definirse en el encabezado del mismo. Cada parámetro debe especificar el tipo de datos con el que se corresponde.

En la invocación del módulo deben definirse también estos parámetros. Aquellos que se definen en el llamado del módulo reciben el nombre de **parámetros actuales**, mientras que los parámetros descritos en el encabezado del módulo invocado (procedimiento o función) se denominan **parámetros formales**.

Una **variable local** es aquella que está declarada y definida dentro de un programa o módulo, en el sentido que está dentro de ese módulo y es distinta a cualquier variable que tenga el mismo nombre y que estuviera declarada en otro lugar del programa.

Una **global** es aquella que está declarada en el programa y que puede ser utilizada por cualquier módulo de este.

Una **estructura de datos** es un conjunto de variables (no necesariamente del mismo tipo) relacionadas entre sí de diversas formas que representa un objeto o elemento de un problema.

Una estructura de datos puede ser homogénea o heterogénea dependiendo de los tipos de datos que componen su estructura, y también se pueden clasificar de acuerdo a la ocupación de memoria estática y dinámica.

Una estructura de datos es **homogénea** si los datos que la componen son todos del mismo tipo, y es **heterogénea** si los datos que la componen son de distinto tipo.

Es **estática** si la cantidad de elementos que contiene es fija, es decir, si la cantidad de memoria que se utiliza no varía durante la ejecución del programa, en tanto es **dinámica** si el número de componentes es variable pues da lugar a que la memoria ocupada cambie a lo largo de la ejecución del programa.

Estructuras estáticas:

Desventaja: el **mal aprovechamiento de la memoria**, ya que si contienen pocos elementos se desperdicia lugar, así como tampoco tienen posibilidad de recibir más elementos de los indicados inicialmente.

Ventaja: su reserva inicial de memoria y **acceso directo a cualquiera de sus elementos**, ya que la posición de cualquiera de ellos puede calcularse como un desplazamiento de la posición inicial de la estructura.

Estructuras dinámicas:

Ventaja: **pueden reservar la memoria necesaria** para cada uno de sus elementos a medida que el algoritmo lo requiera; esto implica un mejor uso de la memoria y la posibilidad de extender las estructuras según las necesidades del problema.

Desventaja: **los datos deben ser accedidos en forma secuencial** dado que sus posiciones no son consecutivas físicamente en la memoria

Los **registros** son uno de los tipos de datos estructurados más utilizados ya que permiten agrupar datos de diferentes clases y con una conexión lógica en una única estructura.

Son un conjunto de valores con tres características básicas:

* Los valores pueden ser de distinto tipo (ósea, los registros son estructuras heterogéneas).
* Los valores almacenados en un registro se llaman campos y cada uno tiene un identificador. Pueden ser accedidos de forma directa, sin necesidad de pasar por todos los anteriores para acceder a uno en específico.
* El almacenamiento ocupado por un registro es fijo (ósea, los registros son estructuras estáticas).

Los **arreglos** son datoscompuestos indexados (ordenados por un índice) que permiten operar sobre cualquier elemento de la estructura especificando su **posición** en la misma

Un **tipo de dato arreglo** (*array*) es una colección ordenada e indexada de elementos, con las siguientes características:

* Todos los elementos son del mismo tipo (es un tipo homogéneo).

2. Los elementos pueden recuperarse en cualquier orden simplemente indicando la posición que ocupan dentro de la estructura (por este motivo es una estructura indexada).

3. La memoria ocupada a lo largo de la ejecución del programa es fija (por ello es una estructura de datos estática).

Una **lista** es un conjunto de elementos de tipo homogéneo donde los mismos no están almacenados de forma secuencial en la memoria; esto significa que, a pesar de mantener un orden lógico interno, los componentes de una lista pueden aparecer físicamente dispersos en la memoria.

Con las estructuras estáticas dadas durante la cursada (registros y arreglos) se vio que una vez dada la declaración de su estructura, queda definida su forma y ubicación en memoria. Con las listas no pasa esto porque son estructuras dinámicas.

Un **puntero** es un tipo de variable en la cual se almacena la dirección de un dato, permitiendo manejar direcciones "apuntando" a un elemento determinado.

|  |  |
| --- | --- |
| NIL | DISPOSE |
| Libera la conexión que existe entre la variable y la posición de memoria.    La memoria sigue ocupada pero no se puede referenciar ni utilizar. | Libera la posición de memoria de forma que la memoria liberada pueda ser utilizada en otro momento en el programa. |

* Diferencia entre Arreglos y Listas:

La principal diferencia es que los arreglos son estructuras estáticas lo cual nos obliga a tener una dimensión física; mientras que las listas son estructuras de datos dinámicas que nos permiten ir agregando o eliminando elementos a lo largo del programa siempre que sea requerido (y la memoria lo permita).  
Los arreglos son una estructura indexada (necesitan un índice para acceder al contenido del arreglo); mientras que las listas son una estructura lineal.  
Las dos estructuras son homogéneas.

Un programa es correcto si cumple con las especificaciones del problema a resolver. Por esta razón es que la especificación debe ser completa, precisa y exenta de ambigüedades.

Para medir esto el programador cuenta con procesos de **verificación y validación.**

1 .Verificación implica comprobar que el programa cumple con las precondiciones y postcondiciones del mismo y corroborar que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado. Las tecnicas de correcion se aplican en la etapa de verificacion.

2. Validación (¿Estamos construyendo el producto concreto?): proceso más general, donde se asegura que el software cumple las expectativas del cliente (hace lo que espera el cliente).

TÉCNICAS PARA MEDIR LA CORRECCIÓN: (las tres son complementarias).

Testing (pruebas): proceso mediante el cual se proveen evidencias convincentes respecto a que el programa hace el trabajo esperado.   
Estas evidencias se proveen mediante un plan de pruebas que implica generar distintos casos y analizar si hay errores o no.

Debugging (depuración): proceso mediante el cual se pueden identificar y corregir errores. Puede involucrar el diseño y aplicación de pruebas adicionales para ubicar y conocer la naturaleza del error.

Walkthrough: consiste en recorrer el programa frente a una audiencia que no tenga ideas preestablecidas sobre el y que esté dispuesta a descubrir errores u omisiones.   
También sucede a menudo que cuando no se puede detectar un error el programador trate de probar que no existe, pero mientras lo hace puede detectar el error o bien puede que el otro lo encuentre.