Este apunte contiene secciones de preguntas teóricas y ejercicios prácticas provistos por la cátedra durante el curso de apoyo para rendir final de CADP, hay algunas preguntas que son para reflexionar y otras más de investigar, yo soy un alumno que escribió las respuestas mientras preparaba el final para usarlo de apunte, lo hice con mucho cuidado pero si tenés dudas sobre algo o no lo entendés investigalo más, suerte en el final!

Resolución de Problemas. Conceptos y definiciones iniciales

¿Qué es la Informática?

Ciencia que estudia la resolución de problemas del mundo real mediante el uso de una computadora.

¿Cuáles son las etapas en la resolución de un problema?

1. Obtener un problema
2. A partir del problema, obtener un modelo
3. Modularizar
4. Realizar el programa
5. Utilizar la computadora

¿Cómo especificarías las tareas involucradas en cada etapa?

- Obtener un problema y hacerse preguntas sobre el
- A partir del problema obtenido hacer un modelo del mismo
- Descomponer el problema en partes para encontrar una solución
- Diseño de su implementación, escribir el programa y elegir los datos a representar
- La computadora es capaz de aceptar datos de entrada, ejecutar con ellos calculos aritméticos y lógicos y dar información de salirda (resueltos), bajo control de un programa previamente almacenado en memoria

¿Qué es un algoritmo? ¿Qué es un programa? ¿Cuáles son las partes que componen un programa?

- Un **algoritmo** es un conjunto de pasos (instrucciones) a realizar sobre un autómata para obtener un resultado deseado en un tiempo finito
- **Programa** = Algoritmos + Datos
- Las instrucciones o acciones representan las operaciones que ejecuta la computadora al interpretar un programa. Un conjunto de instrucciones forma un algoritmo

  +

  Los datos son los valores de información de los que se necesita disponer y a veces transformar para ejecutar la función del programa.

¿Qué es una Precondición y una postcondición? ¿Como ejemplificaría cada una de ellas?

Pre-condición: es la información que se conoce como verdadera antes de iniciar el programa (o módulo)
Por ejemplo, en una función división las precondiciones son que los parámetros son números, y que el divisor sea distinto de 0. Tener una precondición permite asumir desde el código que no es necesario lidiar con los casos en que las precondiciones no se cumplen.

Post-condición: es la información que debería ser verdadera al concluir el programa (o módulo), si se cumplen adecuadamente los pasos especificados
En el ejemplo anterior, la función división con las precondiciones asignadas, puede asegurar que devolverá un número correspondiente al cociente solicitado.

Estructuras de Control

¿A qué llamamos estructura de control?

Estructuras que permiten modificar el flujo de ejecución de las instrucciones de un programa

¿Qué estructuras de control conoces?

Secuencia - Una sucesión de operaciones en la que el orden de ejecución coincide con el orden de aparición físico de las instrucciones
Iteración - Se ejecuta un bloque de acciones 0, 1 o más veces dependiendo de la evaluación de una condición.
Decisión - Toma decisiones en función de datos del problema.

¿Cómo clasificarías las estructuras de control que conoces en repetitivas, iterativas precondicional y postcondicional? ¿Cómo diferencias estos conceptos de estructura de control repetitiva, iterativa precondicional e iterativa postcondicional?

Las iterativas pre condicionales son cuando la condición se evalúa antes de ejecutar el bloque, mientras que la postcondición la ejecución se evalúa después de ejecutar el bloque de acciones.

¿Cómo implementarías la estructura repetitiva utilizando la estructura iterativa precondicional y postcondicional? ¿Hay alguna situación para la que la solución presente dificultades?

Precondicional : Un claro ejemplo es la sentencia While, mientras una condición sea verdadera se ejecuta un bloque de instrucciones.
Post condicional : Por ejemplo, Repeat Until, se ejecuta un bloque y mientras una condición no se cumpla volverá a ejecutarse.

La solución post-condicional presenta dificultad cuando se desea evaluar la condición antes de ejecutar la iteración, dado que el bloque se ejecutará por lo menos una vez antes de evaluar la condición

¿Cómo escribirías una forma equivalente de la estructura de control FOR utilizando la estructura de control WHILE? Y de forma contraria, es decir a partir de WHILE escribir una equivalencia con FOR, como lo harías?

por ejemplo, si quisiera escribir for i:=1 to 10 con la estructura while:
i := 1; while (i <= 10) do begin Accion; i := i +1; end;
No siempre podría escribirse una equivalencia a un while con un for ya que el for requiere saber a priori cuántas veces debe ejecutarse el código, mientras que el while no.

Datos y Tipos de Datos. Alcance de los datos

¿Qué es un dato?

Es una representación de un objeto del mundo real mediante la cual podemos modelizar aspectos del problema que se quiere resolver con un programa sobre una computadora. Puede ser variable o constante

¿Cómo diferencias el concepto de variable y el de constante?

Se diferencian en que las variables pueden cambiar su valor durante el programa y las constantes NO

¿A qué llamamos variables globales y variables locales?

Las variables globales son aquellas que podemos utilizar en todo el programa, incluyendo en los módulos; mientras las variables locales sólo pueden usarse dentro del módulo en que fueron declaradas

Cuáles de los siguientes conceptos se ven afectados cuando se utilizan variables globales:

a) Protección de los datos.

b) Cantidad de memoria utilizada.
c) Tiempo de ejecución del algoritmo.
d) Corrección de programa
e) Cantidad de líneas de código del programa.
f) Mantenimiento posterior del programa.
Justificar en cada caso.

a) Al utilizar variables locales estamos protegiendo al programa de que los módulos no alteren algún valor no deseado.
b) En las variables globales el espacio utilizado suele ser menor que declarar varias variables locales dentro de cada subrutina, por otra parte al utilizar variables locales el espacio se libera al finalizar el módulo
c) No se ve afectado
d) Puede afectarse, al utilizar variables globales en lugar de parámetros se corre el riesgo de alterar valores y que el programa no funcione como debería
e) No habría un cambio notorio
f) Es mucho más fácil mantener un programa que utilice variables locales como corresponde, ya que al hacer cambios en una variable global se debe contemplar que no traiga problemas en todos los módulos

¿Cómo definirías el concepto de tipo de dato? ¿Por qué crees que es útil tener tipos de datos?

Es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos, se caracterizan por:

- un rango de valores posibles
- un conjunto de operaciones permitidas
- una representación interna

Resulta útil para la gestión de la información

¿Qué diferencias hay entre los tipos de datos estándar de un lenguaje y los tipos definidos por el usuario?

El conjunto de valores, operaciones permitidas y representación interna de los tipos de datos estándar de un lenguaje están definidos y acotados por el lenguaje; un aspecto muy importante en los lenguajes de programación es la capacidad de especificar y manejar datos no estándar, indicando valores permitidos, operaciones válidas y representación interna. Un tipo de dato no estándar es aquel que no existe en la definición del lenguaje y el programador es el encargado de su especificación

¿Qué entiendes por ocultamiento y protección de datos? ¿Cómo defines el concepto de alcance de los datos?

Ocultamiento de datos significa que los datos exclusivos de un módulo no deben ser “visibles” o utilizables por los demás módulos.
Protección de datos se refiere a evitar que otros módulos no alteren un valor no deseado
El **alcance** es una propiedad de las **variables**: se refiere a su visibilidad (aquella región del programa donde la **variable** puede utilizarse). Los distintos tipos de **variables**, tienen distintas reglas de **alcance**.

¿Cómo vinculas los tres conceptos anteriores? ¿Podrías detallar cuáles serían los problemas de los lenguajes de programación que no implementen estos conceptos?

Están muy vinculados, el ocultamiento de datos tiene como objetivo proteger los datos y el alcance de ellos es donde pueden utilizarse
Al permitir que los datos puedan ser accedidos desde cualquier módulo se dificulta el trabajo en equipo ya que diferentes programadores tendrían que ponerse de acuerdo en que variables no deben usar en sus módulos y cuales si, además sería difícil utilizarlo y mantenerlo ya que se debería contemplar los efectos que tendría en todo el programa al momento de utilizar una variable

Modularización. Parámetros. Posibilidades en Pascal

¿Cómo defines el concepto de modularización? ¿Cuáles son sus principales características y ventajas?

Modularizar significa dividir el problema en partes funcionalmente independientes, que encapsulan operaciones y datos
Ventajas: Mayor productividad, Usabilidad, Facilidad de mantenimiento, Facilidad de crecimiento, Legibilidad.

¿Cómo diseñarías una solución modularizada adecuadamente? ¿Qué consideraciones tendrías en cuenta al descomponer en módulos?

Separaría el programa en funciones lógicas con datos propios y datos de comunicación perfectamente especificados.

- Cada subproblema está en un mismo nivel de detalle.
- Cada subproblema puede resolverse independientemente
- Las soluciones de los subproblemas pueden combinarse para resolver el problema original

¿Cómo relacionarías la modularización con la reusabilidad y el mantenimiento de los sistemas de software?

La descomposición funcional que nos ofrece la modularización favorece el reuso del software desarrollado.
La división lógica de un sistema en módulos permite aislar los errores que se producen con mayor facilidad, esto significa corregir los errores en menos tiempo y disminuye los costos de mantenimiento del sistema.

¿Cómo defines el concepto de módulo? ¿Qué módulos reconoce Pascal? ¿Qué diferencias hay entre dichos módulos?

Modulo: tarea específica bien definida, se comunican entre si adecuadamente y cooperan para conseguir un objetivo común.
En pascal existen los módulos PROCEDURE y FUNCTION
La diferencia radica en que los Procédure devuelven **0, 1 o más** valores de **cualquier tipo**; mientras las Function devuelve **un único valor** de tipo **simple**

¿Qué ventajas puede tener declarar tipos y variables dentro de los módulos?

- Reusabilidad
- Mantenimiento
- Se liberaría el espacio que ocupa en memoria una vez finalizado el módulo
- Me aseguro de que otros módulos no utilicen las variables

¿Que desventaja le encontrarías a un programa en el que no se han diseñado módulos?

- Si es muy grande sería imposible reutilizarlo
- Difícil de mantener
- Difícil de leer
- ..

¿Todo procedimiento puede ser transformado en una función? ¿Por qué?

No, si bien es posible en algunos, los procedimientos que devuelven más de 1 valor no pueden transformarse en función (estos solo pueden devolver un valor de tipo simple)

¿Toda función puede ser transformada en un procedimiento? ¿Por qué?

Si, porque siempre puede reemplazarse el valor de retorno con un parámetro por referencia

¿A qué llamamos bloque de programa?

Un bloque de programa es un conjunto de instrucciones que se ejecutan una detrás de otra

¿Cuáles son los mecanismos de comunicación entre módulos en Pascal y qué ventajas/desventajas ofrece cada uno? ¿Cuál es el más conveniente desde el punto de vista de la protección de los datos?

Los mecanismos son las variables globales y los parámetros

Las variables globales tienen como ventaja el hecho del espacio en memoria utilizado, el cual suele ser menor que declarar varias variables locales dentro de cada subrutina, sin embargo si hay muchos módulos que utilicen la misma variable puede terminar volviéndose dependientes unos de otros.
Problemas:

- Demasiados idenfiticadores
- No se especifica la comunicación deseada entre módulos
- Conflictos de nombres de identificadores utilizados por diferentes programadores
- Posibilidad de perder la integridad de los datos

Los parámetros favorecen la integridad de datos al declarar algunas variables como locales a un módulo asegurándose que no puedan ser visibles por otros módulos; así se especifican los datos que se comparten para comunicarse con los demás módulos.

Para proteger los datos es conveniente la parametrización, ya que utilizando variables globales puede modificarse datos de una variable involuntariamente en un módulo que luego deberá utilizar otro módulo.

¿A qué llamamos parámetros por valor y referencia?

Los parámetros por valor es llamado parámetro IN y significa que el módulo recibe (sobre una variable local) un valor proveniente de otro módulo o del programa principal
Con el puede realizar operaciones y/o cálculos pero no producirá ningún cambio ni tendrá incidencia fuera del módulo

La comunicación por referencia (out, inout) significa que el módulo recibe el nombre de una variable (referencia a una dirección) conocida en otros módulos del sistema.

Puede operar con ella y su valor original dentro del módulo, y las modificaciones que se produzcan se reflejan en el resto de los módulos que conocen la variable

¿Qué diferencias hay entre un parámetro por referencia y una variable global?

Los parámetros por referencia necesitan ser especificados en el módulo donde se utilizarán; las variables globales se declaran solo en el programa principal.
Los parámetros por referencia sólo pueden ser utilizados en los módulos donde se haya especificado; las variables globales pueden ser accedidas por cualquier módulo.
Los parámetros por referencia son nombrados y referencian a una variable del programa principal mientras las variables globales se invocan con el mismo nombre de la variable para todo el programa.

¿Qué argumento puedes dar para justificar el hecho de que en las funciones todos los parámetros deberían pasarse por valor?

Por definición las funciones devuelven un único valor y este lleva el nombre de la función (en pascal) los valores que reciba para procesar la información no deberían ser devueltos después de modificarse

¿Los parámetros utilizados en el módulo tienen alcance local o global? ¿Por qué?

Tienen alcance local, ya que solo son visibles dentro del módulo donde fueron declarados