Informe de Trabajo Final de Promoción

Declaración de los tipos de datos utilizados en el programa

Tipo de datos declarados (type):	Sintaxis:
subrOpciones = 'A' .. constOPCIONES; subrCategorias = 1 .. constCATEGORIAS;	// Acotamos los valores de las variables para que en caso de que ingresen valores no correspondientes no se tomen en cuenta.
conjCompletadas = set of subrCategorias; conjRespuestas = set of char;	// Declaramos un conjunto para guardar los colores de la cara del diamante.

cadenaPreg = string[238];
cadenaOpci = string[210];
cadenaColores = string[6] ;
	// Utilizamos tres diferentes datos de tipo string con sus respectivas asignaciones de límite de caracteres para optimizar el uso de memoria.

arrOpciones = array [subrOpciones] of cadenaOpci;	// Usamos un arreglo de strings para guardar las opciones de las preguntas
tipoPreguntas = record numCategoria: subrCategorias; pregunta: cadenaPreg; opciones: arrOpciones; respuesta: subrOpciones; explicacion: cadenaOpci; end;	// Creamos un registro para las preguntas con sus respectivas respuestas y/o opciones con sus soluciones

lista = ^nodo;
nodo = record
datos: tipoPreguntas;
sig: lista;
end;
	// Creamos una lista para guardar el registro de las preguntas en donde cada lista creada corresponde a las categorías de la cara del diamante

tipoLista = record pri, ult: lista; end;	// Creamos dos punteros para agregar al final cada pregunta sea mas optimo en tiempo de ejecución
vdlCategorias = array [subrCategorias] of tipoLista;	//Creamos un arreglo de una dimensión para guardar las listas de preguntas de cada categoría de la cara del diamante

Fundamentación de las estructuras elegidas:

Decidimos crear un arreglo de registro ***vdlCategorias*** aprovechandonos de que el arreglo es una estructura homogenea accedida por un indice para guardar una lista de cada categoría donde el elemento es ***tipoLista*** y el índice del arreglo simboliza el número de la categoría, como la cantidad de categorías es conocida y asignada desde el código, como el arreglo es una estructura de acceso directo (ya que para acceder a un valor solo basta con conocer el identificador del campo) nos permite asegurarnos que, siempre y cuando se ingresen bien los datos de cada pregunta, se pueda acceder a todas las preguntas de una misma categoría sin necesidad de recorrer las de otras. Finalmente, como esta estructura se aloja en memoria estática (la cantidad de memoria no varía durante la ejecución del programa) sabemos que siempre que se necesite acceder a las preguntas se puede, permitiéndonos jugar multiples partidas en la misma ejecución del programa, siempre y cuando las estructuras internas al arreglo se mantengan.

El tipo de dato ***tipoLista*** es un registro compuesto por 2 campos del tipo puntero, el puntero ***pri*** nos da la dirección de memoria del primer elemento de una lista de categorías correspondiente que creamos para poder guardar las preguntas a cada categoría y el campo ***ult*** que nos da la dirección de memoria del último elemento de la misma lista.

Decidimos usar estructuras de listas simples para el guardado de las preguntas de cada categoría con el objetivo de asegurar la posibilidad a futuro de poder agregar y cambiar las preguntas disponibles en el juego sin necesidad de modificar el programa, ya que, a diferencia de los arreglos, las listas simples como es una estructura de datos dinámica ya que no dependen de conocer la cantidad de elementos que componen a la estructura por lo tanto la cantidad de memoria varía durante la ejecución y nos permite que para agregar una pregunta más al juego solo basta con agregarla al archivo "***categorias.txt***".

Tanto el campo ***pri*** como ***ult*** está compuesto por un puntero ***lista*** que apunta a un registro ***nodo*** cuyo campo ***sig***, que indica la siguiente posición de la lista y el campo ***datos*** del tipo ***tipoPreguntas***.

El dato ***tipoPreguntas*** es un registro, decidimos utilizar esta estructura de datos heterogenea de acceso directo para unificar toda las caracteristicas de cada pregunta en una solo dato lo que mejora la legibilidad y lectura de las preguntas, a continuacion definimos cada uno de sus campos:

- ***numCategoria*** es un tipo de datos entero que guarda el número de categoría para uso interno del programa (identificar en qué vector de listas se debe guardar cada pregunta).
- ***preguntas*** es un dato de tipo string, donde es la pregunta correspondiente, este string fue acotado a 238 caracteres ya que la pregunta más larga tiene 238 caracteres, pero se puede ajustar fácilmente modificando el type.
- ***opciones*** que es un arreglo de índice ***subrOpciones*** compuesto por elementos del tipo ***cadenaOpci*** (string de 240 caracteres), elegimos usar un arreglo para poder guardar cada pregunta con a que opcion pertenece dentro del mismo dato, como la cantidad de opciones que tiene cada pregunta es conocida (A,B o C) nos pareció adecuado, aunque no permite que cada pregunta tenga una cantidad distinta de opciones.
- ***respuesta*** subrango de caracteres que usamos para guardar la respuesta correcta de cada pregunta.
- ***explicacion*** es un dato de tipo string acotados a 210 caracteres, se encarga de guardar la

respuesta de la pregunta donde es el mismo caso de ***preguntas,*** que debido a que la respuesta con más cantidad de caracteres es de 210 lo acotamos a esa cantidad, pero

- para cambiarlo solo se le debe modificar en el type.

Funcionamiento y diseño del programa

Para el diseño del programa nos basamos en este gráfico, dividimos el funcionamiento del programa en 4 etapas: carga de datos, introducción, partida y cierre del programa.

Carga de datos: donde inicializamos nuestro Vector De Listas (***vdl***), para inicializarlo usamos una variable del tipo ***text*** llamado "***categorias.txt***" que contiene información de todas las preguntas de cada categoría en el siguiente formato:

número de categoría (`1 .. constCATEGORIAS`)
enunciado de pregunta (string[238])
enunciado de opción A, B y C (string[240])
respuesta (char)
enunciado de la solución (string[240])

El módulo ***cargarVDL()*** se encarga de inicializar todos los elementos del vector vdl en nil, abrir el archivo "***categorias.txt***", y separar cada pregunta en un dato ***act*** del tipo pregunta que luego se carga en el vdl. Sabemos que el tiempo de carga del vdl es 6 veces más que el de un archivo de registros (debido a la lectura de cada uno de los campos), pero sabiendo que excede los contenidos de la cátedra decidimos usar un dato text que logramos que funcione.

Introducción: en esta etapa decidimos presentarle la historia y las reglas del juego antes de comenzar la partida del jugador, esta etapa es meramente para brindar una interfaz de usuario mínima.

Partida: esta etapa consiste en un procedimiento especial llamado ***partida()*** al cual le pasamos como parámetro por valor al vector completo de listas, y los parámetros ***resultado*** (variable booleana encargada de devolver si el jugador gana o pierde), ***puntaje*** (variable entera encargada de registrar el total de puntos que consigue el jugador en cada partida), ***error*** (variable entera encargada de contar los intentos del jugador) y ***errordesafio*** (variable entera encargada de contar los la cantidad de errores de la pregunta desafío). Dentro de el modulo partida el jugador elije una categoria (1,2,3,4,5 o el nombre del color de la categoría), lo que llevará a invocar al procedimiento preguntas que se encarga de hacer todas las preguntas de la categoría elegida, hasta que se complete la categoría o hasta que el jugador falle (hasta ***constERRORES***, por defecto 3), donde se le hará elegir al jugador otra categoría. En caso que el jugador falle en la pregunta desafío, se le otorgará la cantidad de intentos definida en ***constERRORDESAFIO*** (2 por defecto). En caso que el jugador pierda o gane se le presentará el resultado y se le dará la opción de jugar de nuevo, reiniciando el proceso ***partida()*** o terminar el programa, que pasará a la siguiente etapa.

Cierre del Programa: una vez que el jugador tomó la decisión de cerrar el juego, se le presentará una pantalla de despedida con el procedimiento ***outro()*** y se invocará el procedimiento ***liberarMemVDL()*** encargado de liberar la memoria utilizada por el VDL.

Preguntas de cada categoría con su respectivas opciones, respuestas y justificaciones

Categoría Roja: "Conceptos introductorios: Datos, Variables"

Pregunta 1: ¿Cuál de estos es un tipo de dato no ordinal?.

A- Entero

B- Real

C- Carácter

Respuesta: El real es un tipo de dato no ordinal ya que tiene infinitos números entre sí, por lo tanto, no tiene un antecesor y un sucesor

Pregunta 2: ¿Qué operación devuelve el cociente entero de una división?.

A- MOD

B- DIV

C- /

Respuesta: La operación DIV es la operación encargada de devolver el cociente de la división entre 2 números.

Pregunta 3: Las constantes se declaran de la siguiente forma:

A- Type NombreConstante = 'Valor';

B- VAR NombreConstante: 'Valor';

C- Const NOMBRECONSTANTE = 'Valor';

Respuesta: En Pascal, las constantes se declaran bajo la palabra reservada const.

Pregunta 4: ¿Cuál de las siguientes opciones es la estructura de control de decisión?.

A- If (condición) then else

B- While (condición) do

C- For i:=1 to valor do

Respuesta: El if, por definición, es la estructura de control de decisión ya que pregunta una vez si una o más condiciones son verdaderas o no, el while es una estructura de control pre condicional y el for es una estructura de control de repetición

Pregunta 5: ¿Cuál de los siguientes tipos de variables no se puede usar como parámetro de la estructura de repetición:

A- Numérico

B- Real

C- Char

Respuesta: El real es el tipo de dato que no se puede utilizar en una variable de control ya que es un tipo de dato no ordinal.

Categoría Verde: "Modularización, comunicación entre módulos, pasaje de parámetros".

Pregunta 1: ¿Qué tipo de datos puede devolver una función?

A- Datos Simples

B- Datos Compuestos

C- Estructuras de datos

Respuesta: Las funciones solo pueden devolver un tipo de dato simple y no puede ser de otro tipo, esto no es aconsejable, ya que las funciones se pueden utilizar dentro de las declaraciones de otras estructuras de control.

Pregunta 2: Para poder modificar los valores de una variable por un procedimiento se recomienda pasarla como:

A- Parámetro por valor

B- Como variable global

C- Parámetro por referencia

Respuesta: Siempre es recomendable pasar las variables como parámetros por referencia si se modifica. Los parámetros pasados por valor no modifican el parámetro real y las variables globales son más difíciles de seguir.

Pregunta 3: ¿Qué es un procedimiento?

A- Un módulo que realiza tareas y puede devolver 0, 1 o más variables a través del método de pasaje por parámetros

B- Un módulo que realiza tareas y solo devuelve un valor

C- Es una estrategia que implica dividir un problema en partes funcionalmente independientes

Respuesta: Un procedimiento no necesariamente debe devolver un solo valor, y la estrategia de subdividir un programa en partes funcionalmente independientes se conoce como modularización.

Pregunta 4: ¿Cómo se invoca una función?

A- Por medio de una estructura de control (asignación, comparación, etc.).

B- Nombre (variable1, variable2, etc);

C- Readln (Nombre (variable1, variable2, etc));

Respuesta: Una función puede ser declarada dentro de cualquier estructura de control que requiera de un dato simple.

Color Azul: Organización de computadoras.

Pregunta 1: Complete con la compuerta lógica correspondiente: 11111....01100 = 01100

A- OR

B- AND

C- XOR

Respuesta: Es AND porqué cuando dos entradas es 1 la compuerta lógica te devuelve 1.

Pregunta 2: Cuáles de estas operaciones es la correcta

A- X XOR 1 = X

B- X AND 0 = X

C- X OR 0 = X

Respuesta: La respuesta es XOR ya que la máscara 1 te devuelve la misma entrada y con AND debe ser con 0 para que te devuelva la misma entrada y con OR basta que una de sus entradas sea 1 para que la función valga 1

Pregunta 3: ¿Cuántas combinaciones de números se pueden realizar con 8 bits en BSS?

A- 0..127

B- 0..255

C- -63...63

Respuesta: Para calcular el rango en BSS es de la siguiente forma 2^n-1 donde n es el número de bits y la respuesta es 255 ya que 2^8-1= 255.

Pregunta 4: La unidad aritmético lógica es la encargada de:

A- Encontrar la celda de memoria en la que se encuentra cada dato requerido.

B- Recibir datos, realizar operaciones lógicas y matemáticas a partir de instrucciones definidas con los mismos y devolver los resultados de dichas operaciones.

C- Recibir resultados de operaciones lógicas y reintegrarse en su respectiva celda de memoria.

Respuesta: Ya que, como lo indica la definición de aritmética: Parte de la matemática que estudia los números y las operaciones que se hacen con ellos.

Pregunta 5: ¿Cómo se realiza la normalización en un sistema de punto flotante con mantisa fraccionaria?

A- Mover el punto fraccionario de manera tal que el bit más significativo después del punto (sin considerar el bit de signo) sea 1 y para no modificar el valor se compensa con modificar el exponente

B- Desplazar en forma dinámica la coma decimal a una posición conveniente y usar el exponente de base 10 para definir la ubicación de la coma

C- Ninguna de las anteriores

Respuesta: La normalización en un sistema de punto flotante con mantisa fraccionaria implica mover el punto fraccionario para que el bit más significativo sea 1, ajustando el exponente para mantener el valor original.

Color Amarillo: Estructuras de datos, Registros.

Pregunta 1: ¿Cuál de estas operaciones se puede aplicar a la variable registro?.

A- Suma

B- Asignación

C- Comparación

Respuesta: Solo se puede realizar la operación asignación en las variables del tipo registro, aunque esto no hable de las operaciones que se le puede aplicar a sus campos.

Pregunta 2: De acuerdo al tipo de datos que lo compone, el registro es una estructura de datos

A- Heterogénea

B- Homogénea

C- Numérica

Respuesta: El tipo de datos registro es una estructura de datos heterogénea ya que sus campos pueden estar compuestos de otros diversos tipos de datos.

Pregunta 3: ¿Cómo le asignamos un valor a un campo de un registro?

A- IdentificadorRegistro,campo := Valor;

B- IdentificadorRegistro.campo1 := Valor;

C- IdentificadorRegistro[campo] := Valor;

Respuesta: Solo se puede realizar la operación asignación directa en un registro de la siguiente forma IdentificadorRegistro.campo1 := Valor;.

Pregunta 4: ¿Cuándo decimos que una estructura de datos es de acceso directo?

A- Cuando se puede acceder a un elemento deseado recorriendo toda la estructura y sumando los elementos adyacentes.

B- Cuando se puede acceder a un elemento deseado recorriendo toda la estructura.

C- Cuando se puede acceder a un elemento deseado sin necesidad de recorrer toda la estructura.

Respuesta: Por definición, las estructuras de datos de acceso directo se pueden acceder a cualquier elemento que la compone sin recorrer la estructura, de ahí su nombre.

Pregunta 5: ¿Qué estructura de control se utiliza para permitir el acceso de todos los campos de un registro directamente?.

A- With .. do

B- Repeat .. Until

C- While .. do

Respuesta: La estructura de control with .. do sirve para acceder directamente a los campos de un registro sin tener que llamar al registro constantemente, solo a sus campos.

Color Morado: Arreglo, Listas y Eficiencia.

Pregunta 1: ¿Qué es un arreglo?

A- Es una colección de elementos homogéneos a los que se accede mediante un índice.

B- Es una colección de elementos homogéneos con una relación lineal que los vincula

C- Es una estructura de datos heterogénea donde a cada dato que lo compone se lo denomina Campo.

Respuesta: El tipo de dato arreglo es una colección de elementos a los que se puede acceder a sus datos mediante un índice.

Pregunta 2: ¿En qué consiste el método de corrección de programas “Walkthrough o Recorrido”?

A- El método consiste en recorrer un programa con alguna otra persona y calcular la cantidad exacta de memoria que ocupa cada variable.

B- El método consiste en recorrer un programa con alguna otra persona.

C- El método consiste en recorrer el programa instrucción por instrucción y compilarlo mentalmente.

Respuesta: Ya que el método walkthrough consiste en ver las instrucciones del programa paso por paso con una persona que no comparta los mismos preconceptos y pensamientos, que a su vez esté dispuesta a descubrir errores.

Pregunta 3: ¿Cuáles de estas operaciones no son permitidas para un tipo de dato puntero?.

A- Readln (puntero);

B- If (puntero1 = puntero2) then

C- Dispose(puntero);

Respuesta: No se puede realizar un Readln sobre un puntero ya que apunta o señala a la variable almacenada en la dirección de memoria que contiene el puntero.

Pregunta 4: ¿Cuando se considera correcto un programa?.

A- Cuando el programa compila.

B- Cuando el programa cumple la función especificada.

C- Cuando el programa está bien modularizado.

Respuesta: Ya que un programa se considera correcto cuando cumple la función que se le propuso, que un programa sea correcto no depende de su calidad, eficiencia o modularización.

Pregunta 5: En caso de perder el primer puntero de una lista:

A- No es posible acceder a la lista, pero la lista sigue cargada en memoria dinámica.

B- La lista automáticamente se libera en la memoria dinámica.

C- La lista automáticamente vuelve al primer nodo y sigue cargada en memoria dinámica.

Respuesta: Cuando no recordamos guardar el primer puntero de una lista se pierde el acceso a la misma ya que la lista es una estructura de datos de acceso secuencial, y no podemos acceder al inicio de la secuencia.

Bibliografía Utilizada:

- “Programación en Turbo Pascal” (originalmente “Programming with Turbo Pascal”) de David W. Carroll, traducido por Laura Magda Figueroa Peña.

- “Programación en Pascal” serie Schaum de Byron S. Gottfried.

- Material de la clase de teoría de la cátedra.

Herramientas Usadas (links):

- Lazarus IDE.

- Github.

- ExcaliDraw (gráficos).

Datos Personales:

Alam Meza:

- Correo: alameza2000@gmail.com
- DNI: 96.223.490

Nicolás Peñalba:

- Correo: nicolaspenalba07@gmail.com
- DNI: 43.738.219