Bueeenas y santas:D.

La verdad es que cada vez tengo más motivación por parte de ustedes, de seguir haciendo este curso. Los aplaudo porque realmente son mis ganas de seguir con ésto. No tengo mucho tiempo libre y generalmente tengo que sacrificar cosas para ponerme a hacer el curso, pero lo hago con gusto, con amor. **Gracias** por la oportunidad que me dan, por haberme elegido:).

Que el conocimiento sea libre, gratuito, universal y objetivo.

Voy a **dividir esta clase en 2** para que me sea más simple enseñarles y que les sea claro. Cualquier cosa, preguntar. Mejor quedar como un idiota una vez, que serlo por siempre.

Sin más vueltas, tenemos que **empezar** con ésto.

Como dije antes, veremos **pseudocódigo y diagrama de flujos** antes de comenzar a pogramar.

Ah... y quiero dar un par de consejos:

Se necesita **metodología y orden** para programar lo más limpio posible.

Esta bueno que vayan aprendiendo **técnicas** de otras personas. Recurrir al código libre es una buena práctica.

No hay que desesperarse para programar, hay que ser **paciente** -el mundo es de ellos-. Se trata de prueba y error, y de errar y volver a tratar.

Voy a intentar de recibir algo de ayuda por parte de compañeros porque cuando se mete el tema programación siempre hay trabajo, y mucho xD. Ya van a odiar y amar programar al mismo tiempo.

“**¿Qué es el pseudocódigo? ¿Algo que es y no es código?”**

Claro, exactamente éso. Mejor explicado: es un tipo de código que no significa ser un lenguaje y por eso no se puede pasar a un compilador. Pero es de gran ayuda para nosotros y para cualquiera, tener una guía con la cual, una vez tenemos lo que queremos hacer y cómo, es fácil pasarla a cualquier código y hacer cambios, o encontrar errores de razonamiento.

Por ejemplo:

> Empieza Programa
>
> **Se muestra un boton de OK**
>
> **Termina programa**

Esto sería un pseudocódigo, pero muy amplio y sin ninguna norma ni regla que nos ayude. Piensen que un programa seguramente lleve tantas líneas que nos vamos a cansar de escribir y pensar.
Vamos con la **primera regla**:

Existen comandos y parámetros. No pueden usarse espacios para escribir cada uno de éstos.

Entonces, nuestro código está mal. Tenemos que arreglarlo.

De nuevo:

Empieza_Programa

Mostrar_botón OK

Termina_Programa

Vamos un poco mejor. Vemos que como no es necesario tener que escribir tanto, a las instrucciones las hice un poco más amenas para leer y más cortas.

Segunda regla:

Cada vez que se inserta un texto, éste va entre comillas.

El OK es un texto, **vamos a cambiarlo**:

Empieza_Programa

Mostrar_botón “OK”

Termina_Programa

Buenísimo, tenemos un pseudocódigo un poco más como la gente. Pero veamos con uno más grande. **El ejercicio**:

Se necesita crear un software que se genere un número aleatorio de 3 cifras y el usuario adivine ese número. El programa le dice si el número que debe adivinar es más alto o más bajo cada vez que intenta adivinar. Si el usuario adivina, se muestra un mensaje con el texto “Le has dado al número”, y el programa se termina.

Aquí tenemos un condicional. Hay acciones que pasan “si” otra pasara. Por ejemplo, el programa ejecutará “el número es más bajo”, sólo si el número que ingresa el usuario es más bajo que el aleatorio. Así que en este caso debemos hacer una pregunta de si el usuario adivinó el número y que pasará en el caso que lo haga y en el caso de que no lo haga.

Bueno muy bien, según las reglas que usamos antes sería:

Empieza_Programa

Generar_Número_Aleatorio

Usuario_Ingresa_Número

¿Adivinó?

Si: Mostrar_Mensaje “Le has dado al número”

No: ¿Es_Mas_Alto?

Si: Mostrar_Mensaje “El número es más alto”

No: Mostrar_Mensaje “El número es más bajo”

Volver

Termina_Programa

“**Esto es inentendible.”**

Es que aún no lo arreglamos. Está muy desarreglado.

Tercer regla:

Cada vez que hay código que es un subcódigo de otro, se debe tabular una vez.

Entonces, Todo el programa interno es subcódigo de Empieza_Programa y Termina_Programa. Lo mismo sucede con las preguntas y sus respectivos Si: y No:. **Reescribamos**, entonces, nuestro código:

Empieza_Programa

Generar_Número_Aleatorio

Usuario_Ingresa_Número

¿Adivinó?

Si: Mostrar_Mensaje “Le has dado al número”

No: ¿Es_Mas_Alto?

Si: Mostrar_Mensaje “El número es más alto”

No: Mostrar_Mensaje “El número es más bajo”

Volver

Termina_Programa

Ahora lo vemos un poco más ordenado. No nos deja confundirnos con los Si: y No:. Pero vamos a poner una **cuarta regla:**

Cuando tenemos más de una instrucción dentro de los “Si:” y “No:” se usará un comando “Empieza” y uno “Termina” para saber de donde a donde va.

Empieza_Programa

Generar_Número_Aleatorio

Usuario_Ingresa_Número

¿Adivinó?

Si: Mostrar_Mensaje “Le has dado al número”

No: Empieza

¿Es_Mas_Alto?

Si: Mostrar_Mensaje “El número es más alto”

No: Mostrar_Mensaje “El número es más bajo”

Volver

Termina

Termina_Programa

Así tenemos en claro que el “**Volver**” sólo se ejecutará en caso de que no haya adivinado el número. En otro caso parece como si se ejecutará de cualquier manera sin importar si le ha dado o no al número y llegaría a realizar una acción que no queremos.

Sigamos con la **quinta regla:**

Debemos específicar a qué lugar regresa el comando “Volver”. Para ésto, usamos una etiqueta que deberá ser una palabra en mayúsculas y encerrada entre corchetes. En el comando Volver se colocará el parámetro correspondiente sin corchetes.

Empieza_Programa

Generar_Número_Aleatorio

[PDR]

Usuario_Ingresa_Número

¿Adivinó?

Si: Mostrar_Mensaje “Le has dado al número”

No: Empieza

¿Es_Mas_Alto?

Si: Mostrar_Mensaje “El número es más alto”

No: Mostrar_Mensaje “El número es más bajo”

Volver [PDR]

Termina

Termina_Programa

Si lo hacemos de esta manera, tenemos bien explicado el código. Aún no terminamos, pero ya falta poco porque vemos que cualquiera puede llegar a entenderlo.

Sexta regla:

Los números deben guardarse con algún nombre. Y cada nombre debe ser único y representa un único valor. Para darle un valor a un nombre, lo hacemos mediante una flecha que indica qué es lo que debemos guardar.

Empieza_Programa

Num_Al ← Generar_Número_Aleatorio

[PDR]

Num_Us ← Usuario_Ingresa_Número

¿Adivinó?

Si: Mostrar_Mensaje “Le has dado al número”

No: Empieza

¿Es_Mas_Alto?

Si: Mostrar_Mensaje “El número es más alto”

No: Mostrar_Mensaje “El número es más bajo”

Volver [PDR]

Termina

Termina_Programa

Y para ya terminar, demos una **última regla** por ahora:

Los condicionales deben ser lo más explicativos posibles.

Empieza_Programa

Num_Al ← Generar_Número_Aleatorio

[PDR]

Num_Us ← Usuario_Ingresa_Número

¿Num_Al = Num_Us?

Si: Mostrar_Mensaje “Le has dado al número”

No: Empieza

¿Num_Al > Num_Us?

Si: Mostrar_Mensaje “El número es más alto”

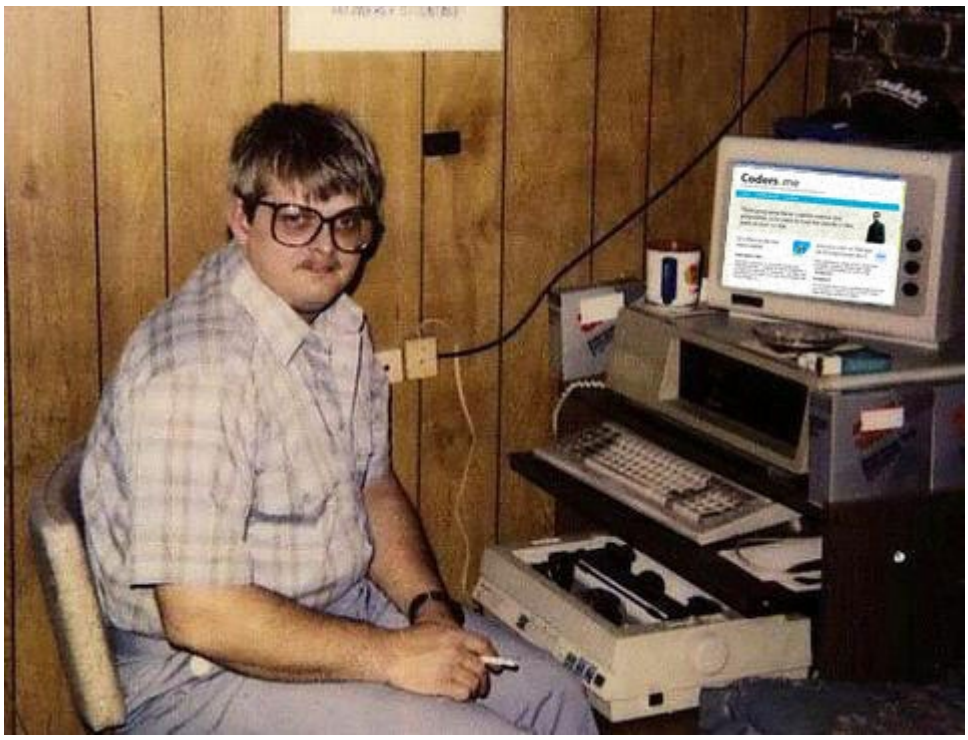
No: Mostrar_Mensaje “El número es más bajo”

Volver [PDR]

Termina

Termina_Programa

Allí vemos que cuando usamos operadores matemáticos para realizar las preguntas, es una buena práctica para simplificar y explicar bien las condiciones.



Por ahora todo muy lindo, pero no vimos nada de diagrama de flujos. Pero lo veremos en la próxima clase ;).

Cualquier cosa pueden mandarme mail a: r0add@hotmail.com

Para donaciones, pueden hacerlo en bitcoin en la dirección siguiente:

1HqpPJbbWJ9H2hAZTmpXnVuoLKkP7RFSvw

Roadd.
