Preparación Olimpiadas Chilenas de Computación

Introducción

Fabio Durán Verdugo
fduran@utalca.cl

David Medina
dmedina10@alumnos.utalca.cl Carlos Campos
ccampos10@alumnos.utalca.cl

31 de mayo de 2014

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Que es la Programación?

¿Que es la Programación?

Una respuesta Básica: Trata de la creación de un algoritmo para entregar una solución a un problema real. Esta solución se realiza bajo un determinado lenguaje de programación.

Aprender a Programar

- Aprender a escribir software.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Que es la Programación?

¿Algoritmos?

Es una secuencia ¸compleja.º con un fin determinado, con un inicio y un fin ordenados de instrucciones que han de seguirse bajo secuencias y condiciones para resolver un problema.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Programas?

¿Programas? (Software)

Es una secuencia lógica de instrucciones que una computadora puede interpretar y ejecutar.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Yo creo...

- Yo creo que todo el mundo debiese saber programar

Yo creo...

- Yo creo que todo el mundo debiese saber programar
- Resolverían sus propios problemas

Yo creo...

- Yo creo que todo el mundo debiese saber programar
- Resolverían sus propios problemas
- Ayudarían a resolver más problemas a otras personas

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Que es la Programación?

El Objetivo General

Resolver problemas básicos a través de la construcción de programas basados en algoritmos, generando acciones hacia la búsqueda de posibles soluciones concretas a propuestas que naceran bajo nuestro interés.

Objetivo Específicos

- Comprender conceptos básicos de computación e informática (programación)

Objetivo Específicos

- Comprender conceptos básicos de computación e informática (programación)
- Construir algoritmos como solución a problemáticas e implementación en un lenguaje de programación (C).

Objetivo Específicos

- Comprender conceptos básicos de computación e informática (programación)
- Construir algoritmos como solución a problemáticas e implementación en un lenguaje de programación (C).
- Cuestionar las soluciones planteadas por nosotros e intentar búscar mejores propuestas.

Ideas a lograr

- Entregar órdenes, ideas, instrucciones precisas en un lenguaje determinado.

Ideas a lograr

- Entregar órdenes, ideas, instrucciones precisas en un lenguaje determinado.
- Desarrollador debe ser inteligente, con criterio, sentido común, y comenzar a generar experiencia.

Ideas a lograr

- Entregar órdenes, ideas, instrucciones precisas en un lenguaje determinado.
- Desarrollador debe ser inteligente, con criterio, sentido común, y comenzar a generar experiencia.
- Computador, es inerte.

Representación de Datos e Información.

** http://desarrollodesistemas.files.wordpress.com/2009/07/datos-informacion1.jpg

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Nuestro gran desafio

Es cómo entender un problema, encontrar una posible solución y traspasar esa solución a un lenguaje que el computador para que la ejecute.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).
  - CPU almacena solo unos pocos calculos que almacena y procesa.

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).
  - CPU almacena solo unos pocos calculos que almacena y procesa.
  - Estos calculos los pueden enviar desde cualquier perisférico.

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).
  - CPU almacena solo unos pocos calculos que almacena y procesa.
  - Estos calculos los pueden enviar desde cualquier perisférico.
  - La CPU es capáz de seguir estos calculos o secuencias de inicio a fin, sin un orden especifico.

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).
  - CPU almacena solo unos pocos calculos que almacena y procesa.
  - Estos calculos los pueden enviar desde cualquier perisférico.
  - La CPU es capáz de seguir estos calculos o secuencias de inicio a fin, sin un orden especifico.
  - Estos calculos o secuencias las llamaremos programas ejecutables.

¿Que es un lenguaje de programación?

- Sabemos que...
  - Un Computador tiene una CPU (Unidad Central de Procesos).
  - CPU almacena solo unos pocos calculos que almacena y procesa.
  - Estos calculos los pueden enviar desde cualquier perisférico.
  - La CPU es capáz de seguir estos calculos o secuencias de inicio a fin, sin un orden especifico.
  - Estos calculos o secuencias las llamaremos programas ejecutables.
  - Los lenguajes de programación se han creado para facilitar la elaboración de programas ejecutables.

Lenguajes de Programación

- Lenguaje artificial

Lenguajes de Programación

- Lenguaje artificial
- Genera instrucciones a una computadora.

Lenguajes de Programación

- Lenguaje artificial
- Genera instrucciones a una computadora.
- Permiten definir tareas de manera más abstractas.
  - Puede que una CPU no tenga la capacidad o forma de resolver una suma a+b=c y tenga que requerir a un sin número de pasos para que obtenga el resultado. En cambio el lenguaje de Programación puede disminuir todo eso entregando solo dos componentes y la forma para obtener el resultado.

Para programar en general se necesita

- Cualquier persona puede escribir un programa.

Para programar en general se necesita

- Cualquier persona puede escribir un programa.
- Escribir una secuencia o algoritmo en un lenguaje de programación (Código Fuente).

Para programar en general se necesita

- Cualquier persona puede escribir un programa.
- Escribir una secuencia o algoritmo en un lenguaje de programación (Código Fuente).
- Existen muchos lenguajes de programación.
  - http://people.ku.edu/ nkinners/LangList/Extras/search.htm

Para programar en general se necesita

- Cualquier persona puede escribir un programa.
- Escribir una secuencia o algoritmo en un lenguaje de programación (Código Fuente).
- Existen muchos lenguajes de programación.
  - http://people.ku.edu/ nkinners/LangList/Extras/search.htm
- Un editor (IDE) para poder escribir.

Para programar en general se necesita

- Cualquier persona puede escribir un programa.
- Escribir una secuencia o algoritmo en un lenguaje de programación (Código Fuente).
- Existen muchos lenguajes de programación.
  - http://people.ku.edu/ nkinners/LangList/Extras/search.htm
- Un editor (IDE) para poder escribir.
- Un compilador o interprete del código escrito para convertir el texto en un programa ejecutable.

```
#Hola Mundo en C
#include <stdio.h>

int main()
{
        printf("Hola mundo");
        return 0;
}
```

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Problema (ecuación de primer grado)

Resolver

$$ax^2 + bx + c = 0$$

Solución

a ——

? x =-b/a

b ——

¿Cuál es el valor de x?

Algoritmo ecuación 1er Grado

```
Variables
  a,b,x de tipo entero
Inicio
    Escribir("Ingrese valores para la ecuacion")
    Leer (a,b)
    Si a = 0 entonces
        escribir ("Error")
    sino
        Inicio
            x= -b/a
            Escribir ("La solucion es": x)
        Fin
    Escribir("Fin programa")
Fin
```

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).
- Elegir Solución .

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).
- Elegir Solución .
- Diseñar la lógica de las solución.
  - Descomponer.
  - Generar tareas de trabajo.
  - Modelar la solución.

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).
- Elegir Solución .
- Diseñar la lógica de las solución.
  - Descomponer.
  - Generar tareas de trabajo.
  - Modelar la solución.
- Implementar la solución.

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).
- Elegir Solución .
- Diseñar la lógica de las solución.
  - Descomponer.
  - Generar tareas de trabajo.
  - Modelar la solución.
- Implementar la solución.
- Testear la implementación.

Resolver problema

Pasos para resolver un problema

- Comprender el problema
  - Concepto.
  - Objetivo.
  - Contexto.
- Búscar soluciones (Google, suena para algo).
- Elegir Solución .
- Diseñar la lógica de las solución.
  - Descomponer.
  - Generar tareas de trabajo.
  - Modelar la solución.
- Implementar la solución.
- Testear la implementación.
- Validar el correcto resultado.

Problemas

¿Que es un problema?

- Situación sobre la cuál se requiere implementar una solución.

Problemas

¿Que es un problema?

- Situación sobre la cuál se requiere implementar una solución.
- Lleva a safisfacer ciertos requerimientos

Enfoque Sistemico

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Datos y Variables

...

- Datos:Objecto simbólicos que representa algo del mundo real.
  - Ejemplo: 31 de mayo de 2014

Datos y Variables

...

- Datos:Objecto simbólicos que representa algo del mundo real.
  - Ejemplo: 31 de mayo de 2014
- Variable: No hace referencia a algo concreto o explicito.
  - Ejemplo: Velocidad promedio de descarga de internet, factores de incremento o disminución.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

¿Que es un Modelos?

Pensando en un Modelo

- Podemos describir como una estructura para dar la razón, abstraer una idea.

¿Que es un Modelos?

Pensando en un Modelo

- Podemos describir como una estructura para dar la razón, abstraer una idea.
- Idealmente debemos generar un inicio y un fin a nuestra idea o posible solución entregando los pasos a seguir y las condiciones que debemos considerar.

¿Que es un Modelos?

Pensando en un Modelo

- Podemos describir como una estructura para dar la razón, abstraer una idea.
- Idealmente debemos generar un inicio y un fin a nuestra idea o posible solución entregando los pasos a seguir y las condiciones que debemos considerar.
- Ejemplo: Un arquitecto entrega un plano de edificación para la creación de un edificio.

¿Que es un Modelos?

Pensando en un Modelo

- Podemos describir como una estructura para dar la razón, abstraer una idea.
- Idealmente debemos generar un inicio y un fin a nuestra idea o posible solución entregando los pasos a seguir y las condiciones que debemos considerar.
- Ejemplo: Un arquitecto entrega un plano de edificación para la creación de un edificio.
- El modelo ideal permite que pueda tener cambios para poder mejorar la solución propuesta. (Experiencia).

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Algoritmos

¿Que es un Algortimo?

- Procedimiento detallado para resolver un problema entregando pasos, condiciones en un tiempo finito.

Algoritmos

¿Que es un Algortimo?

- Procedimiento detallado para resolver un problema entregando pasos, condiciones en un tiempo finito.
- Se construye en base a operaciones básicas que logran controlar la situación y flujo del propio algoritmo.

Algoritmos

¿Que es un Algortimo?

- Procedimiento detallado para resolver un problema entregando pasos, condiciones en un tiempo finito.
- Se construye en base a operaciones básicas que logran controlar la situación y flujo del propio algoritmo.
- Tiene un inicio y fin.

Algoritmos

¿Que es un Algortimo?

- Procedimiento detallado para resolver un problema entregando pasos, condiciones en un tiempo finito.
- Se construye en base a operaciones básicas que logran controlar la situación y flujo del propio algoritmo.
- Tiene un inicio y fin.
- Debe permitir entrada y salidas de "datos.º çondiciones".

¿Como desarrollar un algoritmo?

Para desarrollar un algoritmo debemos tener:

- Imaginación.

¿Como desarrollar un algoritmo?

Para desarrollar un algoritmo debemos tener:

- Imaginación.
- No reinventar la rueda, si la solución ya existe y es efectiva no intentar crear una nueva, si se puede mejorar, no confundir.

¿Como desarrollar un algoritmo?

Para desarrollar un algoritmo debemos tener:

- Imaginación.
- No reinventar la rueda, si la solución ya existe y es efectiva no intentar crear una nueva, si se puede mejorar, no confundir.
- Dividir un problema puede ser siempre una gran idea.

¿Como desarrollar un algoritmo?

Para desarrollar un algoritmo debemos tener:

- Imaginación.
- No reinventar la rueda, si la solución ya existe y es efectiva no intentar crear una nueva, si se puede mejorar, no confundir.
- Dividir un problema puede ser siempre una gran idea.
- La lograr experiencia debes practicar, y ser constante.

¿Como desarrollar un algoritmo?

Para desarrollar un algoritmo debemos tener:

- Imaginación.
- No reinventar la rueda, si la solución ya existe y es efectiva no intentar crear una nueva, si se puede mejorar, no confundir.
- Dividir un problema puede ser siempre una gran idea.
- La lograr experiencia debes practicar, y ser constante.
- El diseño de algoritmos es una rama muy importante en la rama de la computación, y marca diferencias entre los profesionales.

Describiendo un algoritmo

Podemos describir un algoritmo de la siguiente manera:

- Lenguaje natural.

Describiendo un algoritmo

Podemos describir un algoritmo de la siguiente manera:

- Lenguaje natural.
- Pseudo Lenguaje o Pseudo Código.

Describiendo un algoritmo

Podemos describir un algoritmo de la siguiente manera:

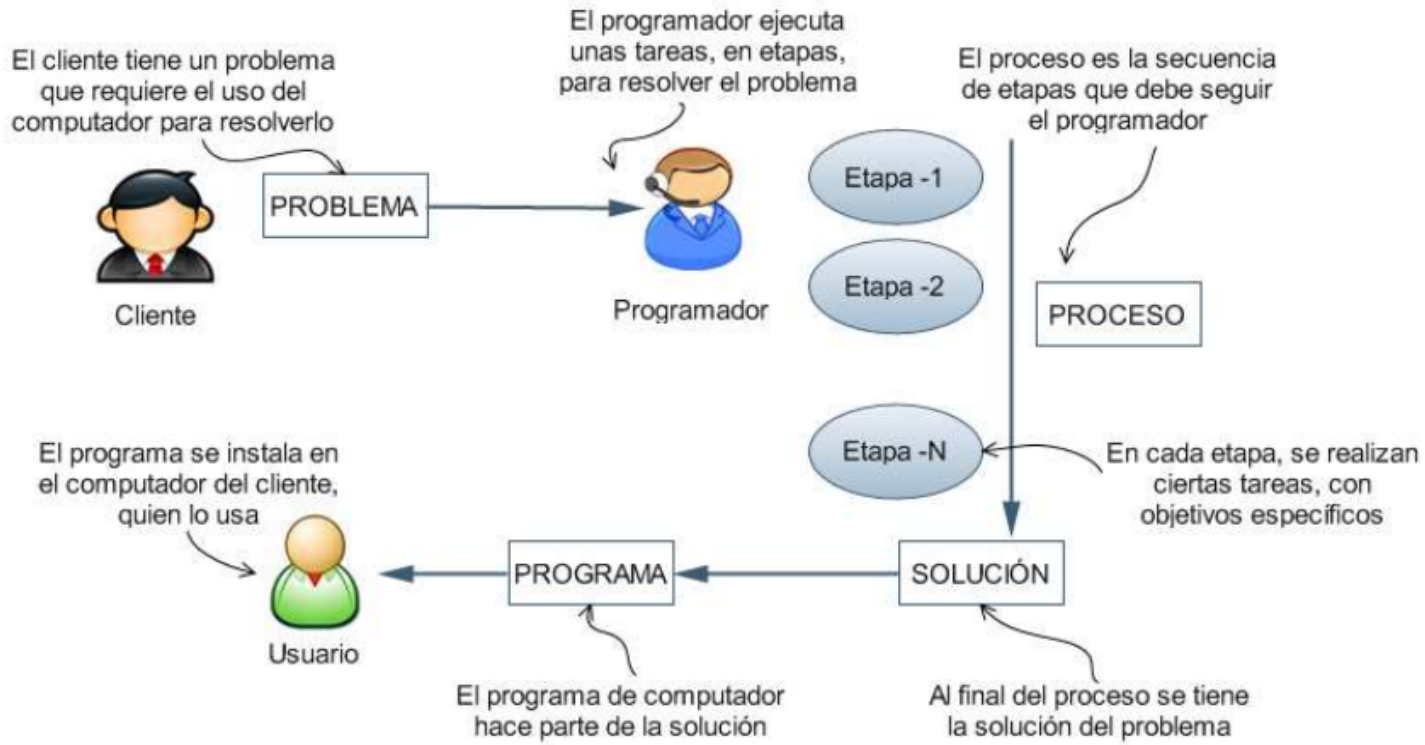
- Lenguaje natural.
- Pseudo Lenguaje o Pseudo Código.
- Lenguaje de programación.

Describiendo un algoritmo

IMPORTANTE

- Un algoritmo no puede ser descrito de forma ambigua.
- Todo el Mundo debe entender la solución.
  - El que planteó el problema
  - El que diseño la solución
  - El que revisa el algoritmo de solución
  - El computador.
  - etc.

Algoritmos



José Antonio Polo - Docente de la Corporación Universitaria Remington

Ejemplo clásico

Máquina de bebidas

Considere que:

- La máquina de bebidas entrega una bebida si depositan 600 pesos.
- La máquina recibe monedas de 10, 50, 100, 500 pesos además de billetes de 1000 y 2000 pesos.
- Si se depositan más de 600 pesos la máquina debe devolver el vuelto que corresponda.
- El proceso inicia cuando alguien deposíta alguna moneda o billete.
- El proceso termina cuando verifíca cuando se debe o no entregar vuelto.

Índice

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
   - Lenguajes de Programación
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Estructuras de condición

En general es una estructura selectiva que, si una cierta condición es verdadera se ejecuta un bloque de instrucciones, si es falsa se ejecuta un bloque diferente de instrucciones.

Si utilizamos un condición es para indicar que según el resultado cierto o falso de una expresión vamos a tomar una decisión para realizar determinadas acciones especificas; seleccionamos las acciones a realizar.

La instrucción a considerar es SI:

Estructuras de Condición

La sintáxis de SI es:

```
SI <condicion> ENTONCES:
    ...
    ...
SINO
    ...
    ...
    ...
FIN SI
```

Estructuras de Condición

Ejemplo simple: Un llamado telefònico:

```
SI <señal ocupado> ENTONCES:
    colgar_telefono()
SINO
    iniciar_conversacion()
FIN SI
```

Estructura de Condición

Ejemplo: Ingrese una vocal y muestra la posición de esa vocal.

```
PSEUDOCOGIGO vocales
VARIABLES vocal: Caracter
INICIO
    ESCRIBIR "Ingrese una vocal":
    LEER vocal
    IF vocal == 'a' O vocal == 'A' ENTONCES:
        ESCRIBIR vocal, "Es la primera vocal"
    FIN SI
    IF vocal == 'e' O vocal == 'E' ENTONCES:
        ESCRIBIR vocal, "Es la segunda vocal"
    FIN SI
    IF vocal == 'i' O vocal == 'I' ENTONCES:
        ESCRIBIR vocal, "Es la tercera vocal"
    FIN SI
    IF vocal == 'o' O vocal == 'O' ENTONCES:
        ESCRIBIR vocal, "Es la cuarta vocal"
    FIN SI
    IF vocall =='u' O vocal == 'U' ENTONCES:
        ESCRIBIR vocal, "Es la quinta vocal"
    FIN SI
FIN
```

Estructuras de Condición

Ejemplo: Leer número y entregar si es positivo o negativo.

```
PSEUDOCODIGO numero_positivo_negativo
VARIABLES numero: real
INICIO
    ESCRIBE "Ingrese un n\'umero cualquiera distinto de cero":
    LEER numero
    SI numero <> 0 ENTONCES:
        SI numero > 0 ENTONCES:
            ESCRIBE "El n\'umero es positivo"
        SINO:
            ESCRIBE "El n\'umero es negativo"
        FIN SI
    SINO
        ESCRIBE "El n\'umero ingresado es cero"
    FIN SI
FIN
```

Estructuras de Condición

Tenemos otra forma de condicionar el problema, es una estructura de condición múltiple que permite evaluar una variable con distintos posibles resultados, ejecutando para cada caso una serie de instrucciones especificas.

EN CASO DE:
Sintáxys:

```
EN CASO DE <expresion> HACER
    CASO <opcion1>
        ....
    CASO <opcion2>
        ....
SINO <expresion no cumple con el rango especificado>
    ....
FIN CASO
```

Estructuras de Condición

Ejemplo: Ingrese un número para identificar un día de la semana, Donde Lunes sea 1 y Domingo sea 7.

```
PSEUDOCODIGO dias_semana
VARIABLE numero: ENTERO
INICIO
    ESCRIBE "Ingresa un numero del 1 al 7"
    LEER numero
    EN CASO DE numero HACER:
        CASO 1: ESCRIBIR "Lunes"
        CASO 2: ESCRIBIR "Martes"
        CASO 3: ESCRIBIR "Miercoles"
        CASO 4: ESCRIBIR "Jueves"
        CASO 5: ESCRIBIR "Viernes"
        CASO 6: ESCRIBIR "Sabado"
        CASO 7: ESCRIBIR "Domingo"
    SINO:
        ESCRIBIR "Escribio un numero fuera del rango de 1 a 7"
    FIN CASO
FIN
```

Índice

- Lenguajes de Programación

1. Introduccíon
   - Introducción
   - ¿Algoritmos?
   - ¿Programas?
   - ¿Por qué es importante saber programación?
   - Objetivos
   - Desafío
2. Resolución de Problemas
   - Problema
   - Datos y Variables
   - Modelos
   - Algoritmos
3. Estructuras
   - Estructuras de Condición
   - Estructuras de Repetición

Estrucuturas de Repetición

La estructura repetitiva se utiliza cuando se quiere que un conjunto de instrucciones se ejecuten un cierto número finito de veces. Llamamos bucle o ciclo a todo proceso que se repite un cierto número de veces dentro de un pseudocódigo o un programa.

Existen dos tipos de estructuras repetitivas:

- Una en donde se tiene establecido el número de veces que un grupo de acciones se van a ejecutar (2, 20, 35, 50 veces), nosotros definimos la iteración.
- El número de repeticiones es desconocido y se hará hasta que se cumpla o no cierta condición.

Estrucuturas de Repetición

Ejemplo: Repetición finita Imprimir diez veces la palabra "Hola Mundo"

```
PSEUDOCODIGO Imprime nombres
VARIABLES i; Entero
INICIO
i=0
PARA i hasta 10:
    imprime "Hola Mundo"
FIN PARA
FIN PROGRAMA
```

Estrucuturas de Repetición

Ejemplo: Repetición infinita (MIENTRAS) Imprimir “Hola Mundo”, hasta presionar la tecla “e”

```
PSEUDOCODIGO Imprime
VARIABLES i; Text
INICIO
i="A"
MIENTRAS i<>’e’ OR i<>’E’:
    IMPRIME "Hola Mundo"
    LEE i
FIN MIENTRAS
FIN PROGRAMA
```

Estrucuturas de Repetición

Ejemplo: Repetición infinita (REPITE) Imprimir “Hola Mundo”, hasta presionar la tecla “e”

```
PSEUDOCODIGO Imprime
VARIABLES i; Text
INICIO
i="A"
REPITE HASTA i<>’e’ OR i<>’E’:
    IMPRIME "Hola Mundo"
    LEE i
FIN REPITE
FIN PROGRAMA
```

Estrucuturas de Repetición

Algunas diferencias entre MIENTRAS y REPITE importantes de considerar.

1. Comprobación de condición:
   - MIENTRAS: La realiza al inicio antes de entrar al bucle.
   - REPITE: La reliza al final, después de entrar al bucle.
2. Las instrucciones (cuerpo)
   - MIENTRAS: Se ejecutan en forma repetitiva si la condición es verdadera.
   - REPITE: Se ejecutan si la condición es falsa
3. Las acciones del bucle:
   - MIENTRAS: Pueden ejecutarse ninguna o más veces
   - REPITE: A lo menos se ejecutan una vez.

Estructuras de Repetición

Ejemplo:

```
EJEMPLO MIENTRAS
LEE password
MIENTRAS password <> 'qwerty':
    ESCRIBE "La contraseña es incorrecta"
FIN MIENTRAS

EJEMPLO REPITE
LEE password
REPITE HASTA password == 'qwerty'
    ESCRIBE "La contraseña es incorrecta"
FIN REPITE
```

Estructuras de Repetición

Cuando podríamos usar las condiciones repetitivas.

- PARA (FOR): Cuando se conozca el limite de la iteración.
- MIENTRAS (WHILE): Dependa de una condición o el valor sea de tipo booleano
- REPITE (REPEAT, DO WHILE WHILE True WHILE False): Çoloquial: Es como Si nosotros encarcelamos a una persona y después preguntamos que hizo, si realmente lo hizo o acaso es culpable o no".

Ejercicios

1. Hacer un pseudocódigo que imprima desde el número 1 hasta el número ingresado por el usuario en pantalla.
2. Hacer un pseudocódigo que imprima solo los pares ingresados desde el 1 hasta el número ingresado por el usuario en pantalla.
3. Realizar un pseudocódigo que solo permite ingresar SI o NO.
4. Crear un pseudocódigo que imprima el mayor y el menor de cinco numeros ingresados por teclado.

Ejercicio 1

Posible Solucion:

```
PSEUDOCODIGO numeros
VARIABLE i, numero: ENTERO
INICIO
    i=0
    IMPRIME 'Ingrese un \'umero'
    LEE numero
    PARA i HASTA numero:
        ESCRIBE i
    FIN PARA
FIN
```

Ejercicio 2

Posible Solucion:

```
PSEUDOCODIGO numero_mientras
VARIABLE i, numero: ENTERO
INICIO
    i=0
    IMPRIME 'Ingrese un \'umero'
    LEE numero
    MIENTRAS i <= numero:
       i=i+2
       ESCRIBE i
    FIN MIENTRAS
FIN
```

Ejercicio 3

Posible Solucion:

```
PSEUDOCODIGO si_no
VARIABLE opcion: Caracter(2)
INICIO
    opcion=''
    MIENTRAS opcion <> 'SI' O opcion <> 'NO':
        ESCRIBIR "Ingrese una opcion SI/NO"
        LEER opcion
    FIN MIENTRAS
FIN
```

Ejercicio 4

Posible Solución:

```
PSEUDOCODIGO numeros_maximos_minimos
VARIABLE maximo, minimo, numero, i, numero_ingresado : Entero
INICIO
    maximo=0
    minimo=9999999999
    numero=0
    i=0
    ESCRIBE "Ingrese el rango de numero a considerar"
    LEE numero_ingresado
    PARA i HASTA numero_ingresado:
        ESCRIBE "Ingrese un numero"
        LEE numero
        IF numero > maximo ENTONCES:
            maximo = numero
        FIN SI
        IF numero < minimo ENTONCES
            minimo = numero
        FIN SI
    FIN PARA
    ESCRIBE "El numero maximo es:", maximo
    ESCRIBE "El numero minimo es:", minimo
FIN
```

¿Preguntas?