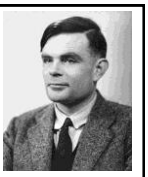
Resolución de Problemas y Algoritmos

Clase 9
Resolución de Problemas con secuencias de datos



Alan M.Turing

Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

Algoritmo

ALGORITMO de Ale para organizar la clase[1]
  Saludar
  Decir: "¿alguna pregunta? "
  MIENTRAS hay preguntas
    -escuchar, pensar y contestar la pregunta
    -decir: " ¿otra pregunta? "
  FIN REPETIR
  REPETIR
    explicar un tema nuevo
  HASTA el fin de la clase o no queden más temas para explicar
  Despedirme

[1]A pesar que este algoritmo puede resultar simpático y muestra el uso de la repetición (*repetir-mientras* y *repetir-hasta)* aplicada a una situación de mi vida cotidiana. Hay una observación importante que no puedo dejar de hacer: la vida en general es impredecible, y no sé si es posible tener un algoritmo que indique como resolver todas las cosas. Una clase como esta no escapa a esa observación. ☺

Conceptos (repaso)

Tipo de Dato: define el conjunto de valores posibles que puede tomar una variable, las operaciones que pueden aplicarse, y cual es la representación interna.

Los tipos de datos en Pascal se pueden dividir en:

- **Simples**
  Ejemplos que vimos en RPA:
  INTEGER, REAL, CHAR, BOOLEAN
- **Estructurados**
  Ejemplos que veremos en RPA:
  FILE OF … (archivos de datos) y TEXT (archivo de texto)

Archivos de datos (repaso)

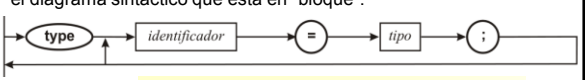
La palabras reservadas FILE y OF se utilizan en Pascal como un constructor de un nuevo tipo de dato (estructurado) creado por el programador sobre la base de otros tipos ya existentes. Esto permite al programador trabajar con archivos de datos. Como se puede ver abajo, el programador puede crear un nuevo tipo de dato archivo y luego declarar variables de ese tipo, como así también directamente declarar variables de tipo archivo.

El identificador reservado TYPE indica que lo que sigue son declaraciones de nuevos tipos de datos creados por el programador.

```
Program ejemplo;
TYPE nuevo_tipo = FILE OF integer;
     archivo_letras = FILE OF char;
VAR  numeros, valores : nuevo_tipo;
     temperaturas, lluvias: FILE OF real;
     letras: archivo_letras;
     decisiones: FILE OF boolean;
```


Tipos creados por el programador (repaso)

En Pascal, la palabra reservada **TYPE** permite al programador crear nuevos tipo de datos, sobre la base de otros tipos ya existentes. Vea el diagrama sintáctico que está en "bloque".



Ejemplo:

```
Program ejemplo;
TYPE nuevo_tipo = FILE OF integer;
VAR valores : nuevo_tipo;
```

Observación importante sobre los archivos declarados con FILE OF: estos archivos no son archivos de texto, son archivos de datos, y por lo tanto no podrá ver o editar su contenido con editores de texto como por ejemplo el "*notepad*". Para disponer de archivos de texto se debe usar el tipo predefinido **TEXT** (que veremos pronto).

Primitivas de Pascal para archivos de datos

- assign(F,N): vincula F con N (nombre del archivo en SO).
- rewrite(F): crea un archivo nuevo con el nombre N que está vinculado al manejador de archivo F (si ya existe otro archivo con ese nombre N se sobre-escribe y se pierde el viejo archivo).
- write(F,e): en un archivo F creado con rewrite, escribe el valor de "e" a continuación del último elemento de F.
- close(F): cierra el archivo vinculado al *manejador* F.
- reset(F): abre un archivo existente de nombre N para leer, y queda preparado para leer el primer elemento.
- read(F,e): lee un elemento del archivo F, copia el valor leído en "e" y queda preparado para leer el siguiente elemento (si existe) o queda en el fin del archivo.
- eof(F) (end of file – fin de archivo): retorna TRUE si se llegó al final de un archivo o FALSE en caso contrario.

Ejemplo propuesto para hacer en el pizarrón

Problema: crear un archivo de números reales llamado "numeros.reales" y permitir al usuario ingresar y almacenar en él tantos elementos en el archivo como quiera.

```
Algoritmo Cargar un archivo
    Crear el archivo.
    Repetir
        Leer un elemento,
        Almacenar en el archivo
        Preguntar si quiere cargar otro número
    Hasta que no quiera cargar más
    Cerrar el archivo.
fin.
```


Parte de la implementación que hicimos en el pizarrón

```
Program  crear_archivo_reales;
TYPE  archivo_reales = FILE OF real; // tipo nuevo creado por el constructor
VAR archi : archivo_reales; // manejador del archivo de reales
      opcion : char; // letra que contendrá la respuesta del usuario
begin
assign(archi, 'números.reales'); // vincula el nombre con el manejador
rewrite(archi); // crea el archivo vacío en una carpeta del disco duro
REPEAT
        … // el resto del programa fue desarrollado en el pizarrón
UNTIL (opcion = 'N') or (opcion = 'n');
close(archi); // el archivo queda cerrado y listo para ser usado por otros
end.
```


Ejemplo

Problema: escriba un programa que busque cuantas veces está el elemento E (ingresado por el usuario) en el archivo "numeros.reales" el cual ya fue creado.

```
Algoritmo contar cuantas veces
    Abrir el archivo para leer
    Solicitar número a buscar
    Asignar el valor inicial 0 a cantidad de veces
    Repetir mientras no sea fin de archivo (EOF)
        Leer un elemento del archivo
        Si el elemento es el buscado
          entonces incrementar cantidad de veces en uno
    Cerrar el archivo.
fin.
```


Parte de la implementación que hicimos en el pizarrón

```
Program  leer_archivo_reales;
TYPE  archivo_nums = FILE OF real; // tipo nuevo creado por el constructor
VAR  numeros: archivo_nums; // manejador del archivo de reales
      buscado : real; // contendrá el nro buscado ingresado por el usuario
      leido: real; //contendrá valores que va leyendo de a uno del archivo
      cantidad: integer; // contendrá la cantidad de apariciones encontradas
begin
assign(numeros, 'números.reales'); // vincula el nombre con el manejador
reset(numeros); // abre el archivo para leer
… // aquí se implementó el ingreso de datos e inicialización
WHILE NOT eof(numeros) DO // mientras no sea el fin del archivo numeros
    begin
        … // el resto del programa fue desarrollado en el pizarrón
    end; // end del while
close(numeros); // el archivo queda cerrado y listo para ser usado por otros
end.
```


Información adicional

Alan M. Turing, (23/6/1912 - 7/6/1954)

- **Matemático, científico de computación, lógico, criptógrafo y filósofo británico.**
- **En 1936 publicó la definición teórica de su "*Universal computing machine*".**
- **En 1940 junto a otros desarrolló la idea de que en la misma memoria de un computadora estén programa y datos.**
- **En 1948 fue nombrado director of Computing Machine Laboratory at the University of Manchester donde se estaba construyendo "Manchester Mark 1" (ver foto siguiente)**

http://es.wikipedia.org/wiki/Alan_Turing

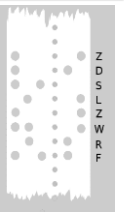
Primeras computadoras: Mark 1

El desarrollo de Mark 1 comenzó en 1948 y se basó en la arquitectura de programa almacenado. Estuvo completamente operable para octubre de 1949.

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 13

Alan Turing y Manchester Mark 1

- **Manchester Mark 1 disponía de una RAM construida con un tubo de rayos catódicos (Williams–Kilburn tube)**
- **Mark 1 no tenía sistema operativo, ni lenguaje ensamblador, los programas debían ser escritos en binario.**
- **Alan Turing desarrolló para Mark 1 un esquema de codificación que permitía que programas y datos sean escritos y leídos de una cinta de papel.**



http://en.wikipedia.org/wiki/Manchester_Mark_1

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 14

Alan M. Turing, y la Inteligencia Artificial

En 1950 contribuyó de forma particular e incluso provocativa a la Inteligencia Artificial con su artículo Computing Machinery and Intelligence, el cual comienza así: I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think"....

Luego propone lo siguiente:

"En lugar de intentar dar estas definiciones, remplazaré la pregunta por otra, que está estrechamente relacionada pero está expresada en términos relativamente sin ambigüedad. La nueva versión del problema puede describirse en términos de un juego al cual llamaremos "juego de la imitación". **(el texto completo está en aquí)**

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 15

Juego de la imitación (Imitation Game) Turing 1950

Se juega con tres personas. Un hombre (A), una mujer (B) y (C) un interrogador (de cualquier sexo). El interrogador se encuentra separado de las otras dos (de manera de no poder verlas u oírlas) a las cuales conoce como X e Y.

El objetivo del juego es que C debe determinar cual de los otros dos es A y cual es B. Durante el juego, C puede hacer todo tipo de preguntas a X e Y, (escritas usando algún medio de comunicación). En el juego, el objetivo de A es engañar a C intentando que piense que es la mujer y el de B ayudar a C. Al final del juego C debe indicar si X es A o X es B.

Ahora hacemos la pregunta, "¿qué pasaría si una máquina toma el lugar de A en el juego?" ¿El interrogador decidiría equivocadamente tan a menudo como cuando en el juego había dos personas?. Esta pregunta remplaza a la original, "¿Pueden las máquinas pensar?".

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 16

Test de Turing

El juego de la imitación dio paso a un test (llamado Turing Test) en el cual una persona P se comunica (sin verlos) con dos entidades A y B, donde una de las dos es una persona y la otra es una máquina. Durante el test, P puede hacer todo tipo de preguntas a B y A, e intentar descubrir cual de las dos es la máquina. Si P no puede distinguir quien de los dos es la máquina, entonces la máquina estará demostrando un comportamiento similar al de una persona.

En 1990 se inició un concurso, el Premio Loebner. El premio está dotado con 100.000 dólares para el programa que pase el test, y un premio de consolación para el mejor programa anual.

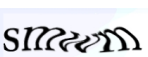
http://es.wikipedia.org/wiki/Test_de_Turing

http://en.wikipedia.org/wiki/Loebner_Prize

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 17

Captcha (o CAPTCHA)

Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar máquinas de humanos). Este test es controlado por una máquina (con un algoritmo público), en lugar de por un humano como en la Prueba de Turing.



Ejemplo: Test CAPTCHA para la secuencia «smwm» que dificulta el reconocimiento OCR por parte de los spambots.

Se trata de una prueba para determinar cuándo el usuario es o no humano. Consiste en que el usuario introduzca correctamente un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina (aún) no es capaz de comprender e introducir la secuencia de forma correcta por lo que solamente un humano podría hacerlo. http://es.wikipedia.org/wiki/Captcha

Volver

Conceptos de Inteligencia Artificial Dr. Alejandro J. García 18