Resolución de Problemas y Algoritmos

Clase 12
Procedimientos y Funciones en Pascal
Resolución de problemas por división y composición: construcción de primitivas.

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

Motivación

Indique a que elemento de Pascal corresponde cada uno de estos identificadores, y agregue un ejemplo más a cada columna de la tabla:

then	integer	WRITE	TRUNC
and	real	READ	EOF
program	char	RESET	CHR

Funciones y procedimientos predefinidos

Al programar en Pascal usamos primitivas predefinidas. Por ejemplo:

```
...
WRITELN ('ingrese un número'); READLN (num);
RESET(F1);  REWRITE(F2);
while not EOF(F1) do begin
  READ (f1,elem);
  if TRUNC(elem) > SQR(num)
   then WRITE (F2, elem)
   else WRITELN ( TRUNC (elem), ' menor que ' , SQR (num));
end;
...
```

expresiones

Algunas Funciones predefinidas

Funciones:
- **Se utilizan en una expresión.**
- **Siempre retornan un valor de un tipo de Pascal.**

Ejemplos de algunas funciones predefinidas :
- **EOF(F): recibe un manejador y retorna boolean**
- **TRUNC(R): recibe real y retorna integer**
- **SQRT(R): recibe real y retorna real**
- **CHR(I): recibe integer y retorna char**

¿Podré construirme mis propias nuevas funciones?

Algunos procedimientos predefinidos

Procedimientos:
- **Se los usa en una sentencia.**
- **Pueden tener 0 o más parámetros.**

Ejemplos de algunos procedimientos predefinidos en Pascal:
- **Writeln**
- **Readln**
- **reset (F)**
- **Rewrite (F)**
- **Assign (F, nombre)**

¿Puedo construir nuevos procedimientos?

Problema propuesto

La potenciación es una operación matemática entre dos términos denominados: base (b) y exponente (e). Se escribe b^e y se lee usualmente como "b elevado a la e".
- **Problema: escriba una función "potencia" que reciba dos enteros positivos (base y exponente) y retorne $base^{exponente}$.**

Solución: Multiplicar el valor de base "exponente" veces

Algoritmo:

resultado:=1;
Repetir exponente veces:
resultado:=resultado*base.

Función Potencia

Algoritmo: potencia
Datos de entrada: base y exponente
Dato de salida: resultado
resultado:=1;
Repetir exponente veces:
resultado:=resultado*base.

Para implementar este algoritmo como una función en Pascal, hay una parte que ya hemos visto:

```
resultado := 1;
FOR i:= 1 TO exponente DO resultado:= resultado*Base;
```

Veremos ahora como indicar el resto de los elementos.

En Pascal toda función tiene:

1. Un nombre (con el cual se la invocará desde una expresión).
2. Parámetros (entre los cuales estarán los datos de entrada).
3. Tipo del resultado (que será el tipo de la función y determinará en que expresión podrá ser usarda)
4. Variables locales (que son propias de la función).
5. Sentencias (también llamado "cuerpo" de la función).
6. Asignación de una expresión al nombre de la función (al menos una vez). Es la forma de retornar un valor.

```
FUNCTION Potencia (Base, Exponente:integer) : integer;
VAR aux,resultado: integer;
BEGIN
  resultado := 1;
  FOR aux:= 1 TO Exponente DO resultado := resultado * Base;
  Potencia:= resultado;
END;
```



```
PROGRAM prueba; {prueba la función potencia}
VAR num1,num2: integer;

FUNCTION Potencia(Base,Exponente:integer) : integer;
VAR aux,resultado: integer;
BEGIN
  resultado := 1;
  FOR aux:= 1 TO Exponente DO
      resultado := resultado * Base;
  Potencia:= resultado;
END;

BEGIN
writeln('ingrese dos enteros positivos);
readln(Num1,Num2);
write(Num1,' a la ',Num2,' es ',potencia(Num1,Num2) );
END.
```

Variables globales
Tipo de la función
Parámetros formales
Variables Locales
Asignación del resultado
Llamada a la función

Funciones en Pascal

Para llamar (invocar o usar) a una función debe:

1) Utilizarse su nombre en una expresión.

2) Coincidir la cantidad de parámetros y el tipo de dato de cada uno de los parámetros.

3) El tipo del resultado debe ser compatible con el tipo de la expresión en la que se lo llama.

Problema propuesto

- Escriba una función "factorial" que reciba un entero positivo N y retorne N! (el factorial de N).

•Escriba ejemplos.
•Escriba la solución.
•Escriba un algoritmo.
•Escriba la función.
•Escriba un programa de prueba.

Para que una función retorne un valor, se debe ejecutar al menos 1 vez, una asignación que en su parte izquierda tenga el nombre de la función y a la derecha una expresión del mismo tipo que la función.

```
FUNCTION Cubo (N:integer):integer;
BEGIN
Cubo:= N*N*N;
END;
```

CORRECTO

```
FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  P := 1;
  FOR aux:= 1 TO 3 DO P := P * N;
  Cubo:= P;
END;
```

CORRECTO

Si no hay una asignación de ese tipo, o ninguna se ejecuta, entonces es un error de programación.

```
FUNCTION Cubo (N:integer):integer;
VAR aux: integer;
BEGIN
aux:= N*N*N;
END;
```

INCORRECTO: falta la asignación de valor a la función.

```
FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  Cubo := 1;
  FOR aux:= 1 TO 3
     DO Cubo := Cubo * N;
END;
```

INCORRECTO: "Cubo" no es una variable, es el nombre de la función. Si usa el nombre de la función en una expresión (como aquí) entonces ¡está llamando a la función!

Invocación a funciones (en expresiones)

```
FUNCTION EsVocal (letra :char): boolean;
BEGIN
   CASE letra OF
    'A','E','I','O','U','a','e','i','o','u': EsVocal:=true;
   ELSE EsVocal:=false;
   END;
END;
```

Algunos ejemplos de invocación:

```
  VAR ch: char;   es_letra_voc:boolean;
  ...
  read(ch);
  es_letra_vocal := EsVocal(ch);
  writeln(EsVocal(ch));
  IF (EsVocal(ch) or (ch='@')) THEN …
  WHILE not EsVocal(ch) and not EOF(archi) DO …
```


Problemas propuestos

- **Escriba un programa que indique cuantas vocales tiene un archivo de caracteres.**
- **Escriba un programa que indique si un archivo tiene al menos una vocal.**

Solución de problemas con primitivas

Hemos visto cómo resolver problemas simples, escribiendo un algoritmo y luego un programa usando asignaciones, condiciones y repeticiones.

En lo que resta de la materia veremos:

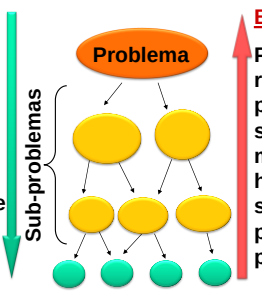
1. **Técnicas para resolver problemas complejos.**
2. **Cómo escribir algoritmos basados en primitivas y algoritmos que son primitivas.**
3. **Cómo implementar en Pascal primitivas y poder usar las dos técnicas anteriores.**

Una primitiva es una operación o acción conocida, utilizada en un algoritmo o programa considerándola como básica.

Conceptos: técnicas "top-down" y "bottom-up"

Top-down:

División del problema principal en subproblemas más simples hasta llegar a problemas que no necesitan dividirse.



Bottom-up:

Por composición, resolviendo primero los subproblemas más simples hasta llegar a solucionar al problema principal.

Ejemplo de trabajo

Problema: Escriba un programa para calcular la suma hasta el término K-ésimo, donde en cada término de posición N, el numerador es 2^N y el denomidador N! Por ejemplo para K = 6

suma = $\underbrace{2/1 + 4/2 + 8/6 + 16/24 + 32/120 + 64/720}_{K=6}$

Solución: sumar (2^N / N!) desde N=1 hasta N=k

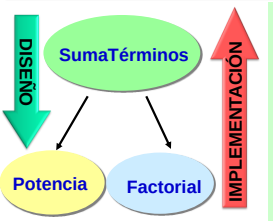
En Pascal, ¿tengo una primitiva para N! (factorial)?

¿tengo una primitiva para 2^N (potencia) ?

Como programador puedo construir nuevas primitivas.

Técnica: Para escribir el algoritmo y el programa se sugiere descomponer el problema en sub-problemas.

Descomposición del problema y primitivas

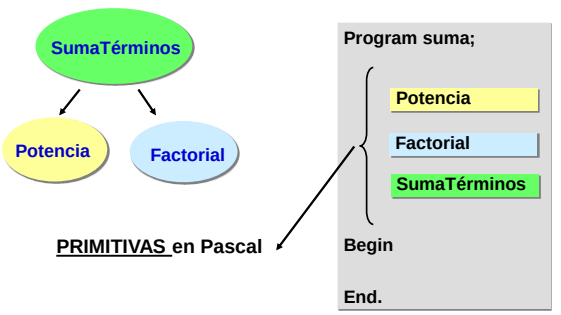



```
PRIMITIVA SumaTerminos
Dato Entrada : K {cant. términos }
Dato Salida : Suma
COMIENZO
  i ← 0     Suma ← 0
  Repetir K veces
    i ← i + 1
    Suma←Suma + Potencia(2,i)
                Factorial(i)
FIN ALGORITMO
```

```
PRIMITIVA Potencia
Datos Entrada : Base, Expo
Datos Salida : Pot
… calcula Base a la Expo…
```

```
PRIMITIVA Factorial
Datos Entrada : N {natural}
Datos Salida : Factorial de N
… calcula factorial de N…
```


Descomposición del problema y primitivas




```
PROGRAM sumaK; {suma los primeros K términos}
VAR K: integer;
FUNCTION FACTORIAL (N:integer): integer;
… completar ….
FUNCTION POTENCIA (B,E:integer):integer;
… completar….
FUNCTION SumaTerminos (tope:integer) : REAL ;
VAR i: integer; suma: real;
BEGIN
  suma:= 0;
  FOR i:=1 TO tope DO
    suma:=suma+ potencia(2,i) / factorial(i) ;
  sumaTerminos:=suma;
END;
BEGIN
writeln('ingrese cantidad de términos'); readln(K);
writeln('la suma es: ', sumaTerminos(K));
END.
```

Variables globales

Parámetros formales

Tipo del Resultado

Variables Locales

Asignación del resultado

Llamada a la función

Tarea (muy importante) propuesta

- Complete el programa suma K y pase en la máquina.
- Realice algunas trazas en papel y luego ejecute para ver como funciona en la máquina.
- Puede poner algunos "writeln" para ver como se van llamando las funciones y como se modifican las variables.
- Por ejemplo:

```
writeln('ingreso a la función potencia');
writeln('salgo de sumaTermino con:', suma, tope);
```