FUNCIONES RECURSIVAS

Curso a cargo de:
Prof. SYLVIA DA ROSA

Trabajo realizado por:
Prof. BEATRIZ FRIONI
Prof. NELIDA GIRALDEZ

T E M A S

1. INDUCCIÓN COMPLETA

Cálculo de cualquier término de una sumatoria.

2. COMBINATORIA

2.1 Definición de Factorial
2.2 Número de combinaciones.
2.3 Número de permutaciones
2.4 Número de arreglos.

3. MÁXIMO COMÚN DIVISOR

Algoritmo de Euclides

4. SUCESIONES DEFINIDAS POR RECURRENCIA

Sucesión de Fibonacci

5. ANEXO

OPERACIONES CON NÚMEROS NATURALES

INTRODUCCIÓN

Se trata de una experiencia de búsqueda y construcción de un conocimiento a partir de un elemento integrador: el concepto de algoritmo. Se eligió trabajar con las definiciones recursivas y utilizar la herramienta computacional para realizar sucesivas pruebas que ayuden a la confirmación de propiedades.
Se utiliza el lenguaje matemático ISETL para integrar elementos de programación a los cursos de matemática discreta para acortar la distancia entre ella y la programación.
ISETL es un lenguaje de programación interactivo, que integra el conjunto de los conocidos como programación matemática, que tiene como característica una sintaxis similar a la notación estándar perteneciente a la matemática discreta.

¿Cuál es nuestro objetivo?

El objetivo es sentar las bases que permitan la elaboración y el desarrollo de un conocimiento en el cual los estudiantes sean capaces de construir conceptos matemáticos apropiados para entender y resolver problemas.
Se busca apreciar la herramienta computacional, en cuanto a su rapidez de respuesta, en un proceso de construcción de conocimiento y realización de actividades imposibles de ejecutarse manualmente por el tiempo y esfuerzo que demandaría la operatoria.
Los algoritmos recursivos se implementan fácilmente en ISETL.

¿Con qué concepción metodológica trabajamos?

Se considera utilizar una metodología de investigación-acción, la cual se fundamenta en una teoría interpretativa de la investigación. El docente actúa como observador participante en el contexto del proyecto.

¿En que concepción didáctica nos basamos?

- La valoración de la importancia de la interacción entre los aspectos cognitivos, psicomotrices y subjetivos que intervienen en la estrategia pedagógica aplicada en este trabajo, se sustenta en las ideas que los diversos educadores introdujeron en las últimas décadas:

  -La concepción del educando como sujeto activo de los procesos educativos.

  -La concepción de la relación interactiva y de diálogo entre el educador y el educando cuyo resultado es el cambio de actitudes, comportamientos y grado de conocimiento de ambos, sin que ello implique la pérdida de sus identidades y roles específicos.

  -La valoración de la importancia de la motivación para obtener aprendizajes significativos y perdurables.

¿Cuál es el rol de la herramienta computacional?

En ese contexto la herramienta computacional juega un rol determinado en relación con el enfoque epistemológico de la matemática. Nos adherimos a lo expresado por G. Polya cuando afirma que:

Las matemáticas son consideradas como una ciencia demostrativa, éste es sólo uno de sus aspectos. La obra matemática se nos presenta, una vez terminada, como puramente demostrativa, consistente en pruebas solamente. No obstante, esta ciencia se asemeja en su desarrollo al de cualquier otro conocimiento humano. Hay que intuir un teorema matemático antes de probarlo. Así como la idea de la prueba antes de llevar a cabo los detalles. Hay que combinar observaciones, seguir analogías y probar una y otra vez. El resultado de la labor demostrativa del matemático es el razonamiento demostrativo, la prueba; pero ésta, a su vez, es construida mediante el razonamiento plausible, mediante la intuición. Si el aprendizaje de la matemática refleja en algún grado la invención de esta ciencia, debe haber en él lugar para la intuición, para la inferencia plausible.

En este trabajo nos interesa el rol de la herramienta computacional, en cuanto a la rapidez de respuesta de la computadora en la búsqueda de ejemplificación de conceptos y verificación de propiedades; este proceso es a su vez fundamental para la construcción de un conocimiento significativo.

Un algoritmo es una descripción precisa de un proceso que termina con la respuesta correcta para cada instancia particular de un problema.

Se plantea el trabajo considerando que:

- La metodología se basa en la participación activa del estudiante como agente de su aprendizaje procurando apoyarlo con el uso del programa informático.
- Las pautas de trabajo son las siguientes:
  Los alumnos trabajando en un laboratorio de computación (dos o tres por máquina), en el tiempo asignado a clases prácticas.
- Los alumnos no tienen nociones específicas sobre el lenguaje ISETL ni programación.
- En un comienzo se utilizaría la computadora realizando ejercicios para resolver trabajos prácticos, luego se trataría de que ellos llegaran a realizar los programas que fueran necesarios para resolver determinadas situaciones.

Buscamos evaluar:

a) El alumno frente a la computadora.

- Capacidad de relación entre simbología matemática y lenguaje computacional.
- Grado de captación de la lógica del sistema.
- Reacciones frente a las respuestas imprevistas de la computadora.

b) El alumno frente al aprendizaje.

- Capacidad de exploración de la validez de propiedades no conocidas por analogía o por intuición.
- Capacidad de inducción de nuevos conocimientos a partir del error.
- Capacidad de demostración en pequeños problemas teóricos.
- Capacidad de planteamiento y solución de un problema de matemática discreta.
- Criterios para el análisis de soluciones.

Definir una función recursivamente consiste en:

- Dar la definición de la función para algún/os elemento/s del dominio, que es el ***caso/s base***.
- Definir la función para un término general del dominio, distinto del caso base, en relación con algún/os término/s anterior/es, que es el ***caso/s inductivo/s***.

1. INDUCCIÓN COMPLETA

En la secuencia de los trabajos a realizar, el primero es:

SUMA DE LOS PRIMEROS NUMEROS NATURALES.

Se llama suma de varios números dados en un cierto orden, al número que se obtiene de la siguiente forma: a la suma de los dos primeros se le agrega el tercer número, a este resultado se le suma el cuarto y así sucesivamente hasta llegar al último número.

0+1+2+3+4+5=[{[(0+1)+2] +3} +4] +5

Queremos calcular la suma de los ***n*** primeros números naturales:

$$0+1+2+3+.....+n-1+n \quad \forall n \in N$$

Por lo tanto:

Le damos un nombre a la suma de los primeros naturales:

$$0+1+2+3+.....+n-1+n = sumo(n)$$

$$\forall n \in N$$

y observamos que:

$$0 + 1 + 2 + 3 + \ldots + n - 1 + n = sumo(n)$$
$$sumo(n-1)$$

sumo(n) = sumo(n-1) + n

Vamos a calcular ***sumo(3):***

n= 3	***sumo(3)***	**i)** ***sumo(2) + 3***
		ii) ***sumo(1) + 2 + 3***
		iii) ***sumo(0) +1 + 2 + 3***
		iv) ***sumo(-1) + 0 + 1 + 2 + 3***
		v) ***sumo(-2) + -1 + 0 + 1 + 2 + 3***
		y así indefinidamente

De lo antedicho podemos deducir que ***sumo(n) = sumo(n-1) + n*** tiene sentido:

$$\forall\, n \geq 1$$

Siendo imprescindible definir previamente: ***sumo(0) = 0***

Por lo tanto:

$$sumo(n) = \begin{cases} 0 & \text{si} \quad n = 0 \\ sumo(n-1) + n & \forall\, n \geq 1 \end{cases}$$

Con esta definición recursiva tratamos ahora de crear un programa en ISETL, con el cual poder hallar cualquier suma de números naturales.

Es necesario previamente explicar la sintaxis del programa, y que los alumnos puedan crear algo similar a:

```
sumo := func (n);
     if n=0 then return 0;
     else return sumo(n-1) + n;
     end;
end;
```

Luego de realizado, se prueba el programa, con la suma de los 100 primeros números naturales.

Digitamos:

>***sumo***(100);

y obtenemos

>5050

Debemos hacerles notar que el programa funciona en forma inversa a cómo nosotros lo haríamos:

Para	Escribimos	El programa hace:	Resultado:
n = 0	***sumo(0)***	***0***	***0***
n = 1	***sumo(1)***	***i) sumo(0) + 1*** ***ii) 0 + 1***	***1***
n = 2	***sumo(2)***	***i) sumo(1) + 2*** ***ii) sumo(0) + 1 + 2*** ***iii) 0 + 1 +2***	***3***
n = 3	***sumo(3)***	**vi)** ***sumo(2) + 3*** **vii)** ***sumo(1) + 2 + 3*** **viii)** ***sumo(0) +1 + 2 + 3*** ***iv) 0 + 1 + 2 + 3***	***6***
n = 4	***sumo(4)***	***i) sumo(3) + 4*** ***ii) sumo(2) + 3 + 4*** ***iii) sumo(1) +2 + 3 + 4*** ***iv) sumo(0) + 1 + 2 + 3 + 4*** ***v) 0 + 1 + 2 + 3 + 4***	***10***

Se observa claramente el proceso de recursión: o sea ***ir hacia atrás.***

Este método o algoritmo empleado es recursivo, es decir se basa en solucionar el problema para un caso concreto, el ***caso base*** y luego para un término general del dominio, distinto del caso base, en relación con algún término anterior, que es el ***caso inductivo***.

Para abreviar la escritura de la suma ahora se puede mencionar que existe otra notación que es utilizando el símbolo ***sumatoria*** representado por la letra griega sigma $\sum$ que fue incorporada al lenguaje matemático en 1755 por Euler:

$$0 + 1 + 2 + 3 + \cdots + n - 1 + n = \sum_{0}^{n} i$$

$$\sum_{0}^{n-1} i$$

Y podemos hacer la definición del símbolo $\sum$ por recurrencia:

Definición:

f es una función de dominio D={i; $i \in$ N, $k \leq i \leq$ n, $k \in$ N, $n \in$ N}

$$\sum_{i=k}^{i=n} f(i) = \begin{cases} f(k) & n = k \\ \sum_{i=k}^{i=n-1} f(i) + f(n) & (\forall n), n \in N, \quad n > k \end{cases}$$

$$sumo(n-1) + n = sumo(n)$$

$$\sum_{0}^{n-1} i \qquad \sum_{0}^{n} i$$

donde: $\sum_{0}^{n} i = \sum_{0}^{n-1} i + n$ expresa lo mismo que lo anterior.

Atención:

Debemos destacar que el programa no termina nunca si le introducimos en el valor de ***n*** un número no natural, pues en esos casos no se llega al caso base.

Por ejemplo si ponemos **–2**

n = -2	**sumo(-2)**	**i)**	**sumo(-3)+(-2)**	
		ii)	**sumo (-4)+(-3)+(-2)**	
		iii)	**.....**	**no termina nunca**

Si no se definió el ***caso base*** el programa continúa indefinidamente, por consiguiente la recursividad no termina.
Esto se debe a que la función está mal definida pues no hemos indicado el dominio y el codominio de la misma.

Por lo tanto es fundamental definir bien la función.

De aquí surge la necesidad de limitar el dominio:

Modificación necesaria del programa;

Definición:

$$S : N \to N$$
$$S(n) = \begin{cases} 0 & si \quad n = 0 \\ S(n-1) + n & \forall\, n \geq 1 \end{cases}$$

El programa de Isetl quedaría de esta forma:

```
$Suma de los primeros números naturales
S :=func(n);
    if (is_integer(n) and n>=0) then
                                  if n=0 then return 0;
                                 else return S(n-1) + n;
                                   end;
    end;
end;
```

Consideramos conveniente crear previamente una función que incluiríamos en ISETL como predefinida para poder limitar el dominio al conjunto de los números naturales:

```
$ números naturales
isnat :=func(n);
          return is_integer(n) and n>=0;
end;
```

De esta modo sería más simple de realizar:

```
$Suma de los primeros números naturales
```

```
S:=func(n);
     if isnat(n)  then
                  if n=0 then return 0;
                  else return S(n-1) + n;
                   end;
     end;
end;
```

Ahora es importante que se pruebe el programa para números tales como: 3, ½ , –3 y que observen que en estos dos últimos casos el programa responde **OM** que para él significa **indefinido**
Explicarles que esto significa que el número introducido es un número que no reconoce y que no pertenece al dominio.

Se podría también poner un mensaje aclaratorio para que el programa informara que estos valores no los reconoce.

```
$Suma de los primeros números naturales
S' :=func(n);
     if isnat(n)  then
                   if n=0 then return 0;
                  else return S'(n-1) + n;
                   end;
     else print "no pertenece al dominio";
      end;
end;
```

En este caso, si introducimos un número no natural además del mensaje de error también nos responde OM.

Se debe insistir en que el resultado siempre es único para un mismo argumento (en este caso OM), y la impresión del mensaje es una acción más que el programa permite hacer para obtener más información.

El programa computa una función y además puede hacer otras cosas.

Hay otros métodos o algoritmos para resolver el mismo problema, por ejemplo el que utilizaremos a continuación:

Suma de los n+1 primeros números naturales

Se sugiere ahora deducir de otra manera la suma de los n+1 primeros números naturales. Por ejemplo, observando que la suma del primero y el último es igual a la del segundo más el penúltimo y así sucesivamente, que todos los equidistantes de los extremos suman **n.**
Por lo tanto:

$$\underset{1+n=n}{\underbrace{0+1}}+\underset{}{2+3+\ldots\ldots+n-1}+\underset{0+n=n}{\underbrace{n}}=\frac{n(n+1)}{2}$$

Y obtenemos así:

$$0+1+2+3+\ldots\ldots+n-1+n=\frac{n(n+1)}{2}$$

$$\underset{suma(n)}{\underbrace{0+1+2+3+\ldots\ldots+n-1+n}}=\frac{n(n+1)}{2}$$

$$\boxed{suma(n)=\frac{n(n+1)}{2}}$$

Por ej. cuando n=5

$$suma(5)=\frac{5.6}{2}=15$$

¿Ésta igualdad será válida para todo n ?

Lo probaremos por Inducción Completa:

$$suma(n) = \frac{n(n+1)}{2} \quad \forall n \in N$$

$$n = 0 \quad \Rightarrow \quad suma(0) = \frac{0(0+1)}{2}$$

H) $$suma(h) = \frac{h(h+1)}{2}$$

T) $$suma(h+1) = \frac{(h+1)(h+2)}{2}$$

Demostración:

$$suma(h+1) = suma(h) + (h+1)$$

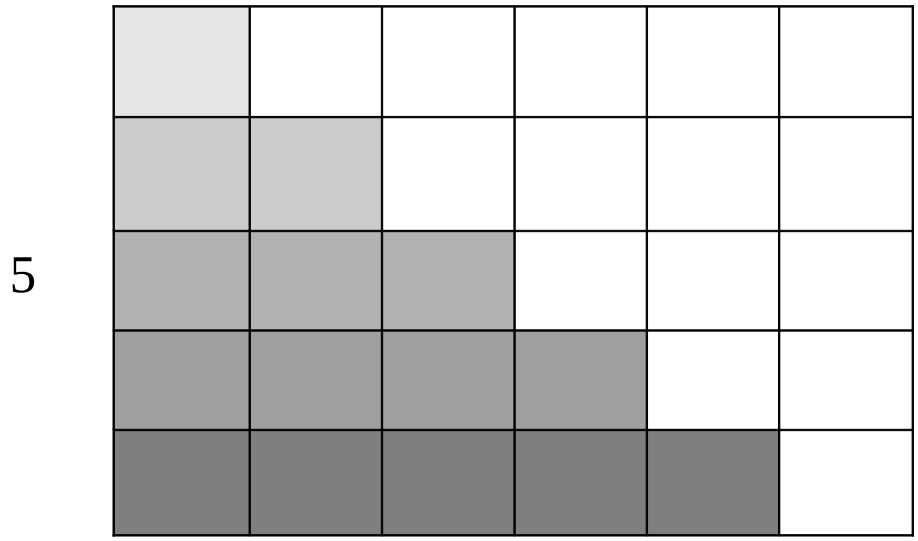
$$suma(h+1) = \frac{h(h+1)}{2} + (h+1)$$

$$suma(h+1) = \frac{h(h+1) + 2(h+1)}{2}$$

$$suma(h+1) = \frac{(h+1)(h+2)}{2}$$

Otra manera de deducir la propiedad puede ser geométricamente. Para ello utilizamos los números triangulares tema que fue desarrollado por los griegos, quienes estudiaron que números se podían representar en forma geométrica. Estos números pueden definirse de una manera puramente geométrica:

En figura tación ue el
segundo 1+2, +2+3 y La r
número suce ne ent a fila
contiene un e que la ngula
Vemos entonces que el número t es la s
números naturales. Vamos a enc para
descubrirla vamos a usar regiones cuadradas en lugar de
calculamos como área de un rectángulo que tiene por eje
largo 5+1. La suma de los primeros 5 números naturales es la mitad del área del
rectángulo o sea $\frac{\mathbf{5\times 6}}{\mathbf{2}}$ que se corresponde con el área sombreada.



Hasta ahora hemos obtenido dos algoritmos distintos para resolver el mismo problema que llamamos ***sumo*** y ***suma***. Ahora queremos comprobar que aplicando cualquiera de ellos llegamos a los mismos resultados.

Por ejemplo:

Probamos en la computadora con el algoritmo recursivo con cualquier número, Ej. 60

El computador nos da como resultado***: sumo***(60)=183

Y luego comprobamos que $\boldsymbol{suma(60) = 60\ \frac{60.61}{2} = 183}$

Observamos que la computadora calcula muy rápidamente cualquier suma finita y nosotros lo podemos comprobar con el segundo algoritmo, por lo tanto:

$\boldsymbol{sumo\ (n) = suma\ (n)}$

¿Esta igualdad será válida cualquiera sea el n$\in \boldsymbol{N}$?

Lo probaremos por Inducción Completa:

$\boldsymbol{n = 0 \ \Rightarrow\ sumo\,(0) = suma\,(0) \ \Rightarrow\ 0 = 0}$

H) $\boldsymbol{n = h \ \Rightarrow\ sumo\,(h) = suma\,(h)}$

T) $\boldsymbol{n = h + 1 \ \Rightarrow\ sumo\,(h+1) = suma\,(h+1)}$

Demostración:

$$\boldsymbol{sumo\,(h+1) = sumo\,(h) + h + 1 = suma\,(h) + h + 1 = \frac{h\,(h+1)}{2} + h + 1 =}$$

$$\boldsymbol{= \frac{(h+1)(h+2)}{2} = suma\,(h+1)}$$

↓ **def. *sumo*** ↓ **H** ↓ **def. *suma*** ↑ **def. *suma***

Observación:

Se podría implementar en ISETL un programa que nos permita calcular ***suma (n)***

```
$Suma de los primeros números naturales
  suma :=func(n);
        if isnat(n)  then
                    if n=0 then return 0;
```

```
                    else return n(n+1)/2;
                end;
        end;
    end;
```

Es decir es importante destacar que siempre hay dos o más formas de resolver el mismo problema

Vamos a realizar otro ejercicio:

suma de los primeros n números impares

Este problema se trata de la misma forma que el anterior, primero estudiando el algoritmo recursivo para solucionarlo, definiendo la función por recurrencia y finalmente el programa. También se presenta otro algoritmo que calcula la suma de los números impares y por último se demuestra la equivalencia de ambas definiciones por inducción completa.

1 + 3 + 5 + ... + 2(n-1)-1 + 2n –1 = *impar(n)*

impar(n) = impar(n-1) +2n-1 $\forall n \geq 2$

impar(1)=1

Definición:

$$impar : N^* \rightarrow N$$

$$impar(n) = \begin{cases} 1 & si \quad n = 1 \\ impar(n-1) + 2n - 1 & \forall n > 1 \end{cases}$$

Realizamos un programa para calcularlo:

```
impar := func(n);
        if isnat(n) and n/=0 then
                                if n=1 then return 1;
                               else return impar(n-1) + 2*n-1;
                                end;
        end;
end;
```

Razonamos como el computador realiza el proceso:

Para	Escribimos	El programa hace	Resultado:
n = 1	impar(1)	1	1
n = 2	impar(2)	i) impar(1) + 3 ii) 1 + 3	4
n = 3	impar(3)	i) impar(2) + 5 ii) impar(1) + 3 + 5 iii) 1 + 3 + 5	9
n = 4	impar(4)	i) impar(3) + 7 ii) impar(2) + 5 + 7 iii) impar(1) +3 + 5 + 7 iv) 1 + 3 + 5 + 7	16

Ahora vamos a deducir la suma de los primeros n impares de otra forma:

$1 = 1 = 1^2$

$1 + 3 + 5 = 9 = 3^2$

$1 + 3 = 4 = 2^2$

$1 + 3 + 5 + 7 + 9 = 25 = 5^2$

$1 + 3 + 5 + 7 = 16 = 4^2$

.....

Intentamos que descubran que la suma tiene por resultado, una potencia de base natural y exponente 2, cuya base es el número de términos sumados o de otra forma que es el (primero + último) sobre 2:

$$\frac{1+3}{2}=2$$
$$\frac{1+5}{2}=3$$
$$\frac{1+7}{2}=4$$

....
....

$$1 + 3 + 5 + \quad + (2n - 1) = \left(\frac{1+(2n-1)}{2}\right)^2 = n^2$$

$$imp(n)=n^2 \quad \forall n \in N^*$$

Otra forma de deducirlo sería geométricamente aplicando los números cuadrados que también fueron estudiados por los griegos.

Se puede observar que un cuadrado se puede partir o dividir en cuadrados menores utilizando una escuadra de carpintero llamada gnomo, si cada vez que ponemos una escuadra agregamos dos puntos podemos ver que con la 1ª escuadra tenemos un punto y considerando

que el lado del cuadrado es igual a 1, con una nueva división tenemos 1+3 puntos y el lado del cuadrado es 2 y $2^2=4$ en el tercer caso vemos que $1+3+5=3^2=9$ y en el 4º caso tenemos que $1+3+5+7=4^2=16$ y así podríamos continuar.
Generalizando vemos que el n-ésimo número cuadrado, n^2, es igual a la suma de los n primeros números impares.

$$\mathbf{1+3+5+7+.....+(2n-1)=n^2}$$

Los antiguos llamaban a los números impares números gnomónicos.

Siguiendo el mismo razonamiento que en el ejercicio anterior solicitamos la demostración por inducción completa.

Podemos probar que con cualquier número el programa responde en forma inmediata. Ej 60

El resultado que da la computadora es 3600

Lo podemos verificar con ***imp(60)***: $60^2=3600$

Y se demostraría por inducción completa que: ***impar(n)= imp(n)***

<u>Análogamente se realiza otro ejercicio:</u>

$$2^0 + 2^1 + 2^2 + ... + 2^{n-1} + 2^n = sum(n)$$

$$\underbrace{2^0 + 2^1 + 2^2 + \cdots + 2^{n-1}}_{sum(n-1)} + 2^n = sum(n)$$

$$sum(n) = sum(n-1) + 2^n \quad \forall\, n \geq 1$$
$$sum(0) = 1$$

Definición:

$$sum: N \rightarrow N$$

$$sum(n) = \begin{cases} 1 & \text{si } n = 0 \\ sum(n-1) + 2^n & \forall n > 1 \end{cases}$$

Programa en ISETL:

```
sum := func(n);
        if isnat(n) then
                if n=0 then return 1;
                else return sum(n-1) + 2**n;
                end;
        end;
end;
```

Probamos el programa:

Para	Escribimos	El programa hace	Resultado:
n = 0	sum(0)	1	1
n = 1	sum(1)	i) sum(0) + 2^1	
		ii) 1 + 3	3
n = 2	sum(2)	i) sum(1) + 2^2	
		ii) sum(0) + 2 + 4	
		iii) 1 + 2 + 4	7
n = 3	sum(3)	i) sum(2) + 2^3	
		ii) sum(1) + 2^2 + 2^3	
		iii) sum(0) +21 + 22 + 23	
		iv) 1 + 2 + 4 + 8	15

Cuando se comprende como procede el programa queremos que lo deduzcan de otra forma:

$2^0 = 2^1-1$
$2^0 +2^1 = 2^2-1$
$2^0 +2^1 +2^2 = 2^3-1$
....
....

$2^0 +2^1 +2^2 +....2^n = 2^{n+1}-1$

$$sum(n) = 2^{n+1} - 1 \qquad \forall n \in N$$

Luego podemos probarlo con cualquier número tan grande como queramos. Ej.14

El resultado que responde la máquina: 32767

Y luego lo verificamos: $2^{14}-1= 32767$

Conclusiones

1º- Todos los ejercicios planteados se resuelven siguiendo dos métodos o algoritmos distintos. Uno de ellos es recursivo, en él: se define el caso base y la solución del caso n se basa en la del caso anterior (que para n es n-1) más un término que es el que varía según el problema. Aquí es donde el elemento integrador del algoritmo tiene su importancia. En estos casos podemos notar que siempre:

f(elemento)=valor
f(n)=f(n-1) + término general

2º- Se probó por inducción completa la equivalencia de ambos algoritmos. Por lo tanto podemos ver que se pueden encontrar varios algoritmos distintos para resolver el mismo problema. Luego se integran los conceptos por inducción completa como método de demostración de la equivalencia. Para cada algoritmo podemos definir funciones matemáticas y además introducimos una herramienta informática como es el lenguaje ISETL que nos permite traducir fácilmente estas definiciones matemáticas a programas con los cuales podemos verificar rápidamente los resultados y también asimilar conceptos. Como por ejemplo, que significa aplicar una función a un elemento que no pertenece al dominio, en este caso el programa responderá OM que significa ***indefinido*** en este lenguaje.

3º- El uso del computador facilita los cálculos y nos permite afianzar los conceptos matemáticos.

2. COMBINATORIA

2.1 FACTORIAL DE UN NÚMERO NATURAL

Factorial de un número es el producto de todos los naturales desde 1 hasta n para $n \geq 2$

n! = 1x 2 x 3 x 4 x .....x (n-1) x n

Definición:

$$\mathbf{fac} : \mathbf{N} \rightarrow \mathbf{N}$$

$$\mathbf{fac} = \begin{cases} \mathbf{1} & si \quad n = 0 \\ (n-1)! \, . n & (\forall n), n \in N, \quad n \geq 1 \end{cases}$$

Programa en ISETL:

```
$factorial de n
fac := func (n);
      if n=0  then return 1;
              else return fac(n-1)*n;
              end;
end;
```

Limitando a n natural e incluyendo un mensaje de error si n no pertenece al dominio:

```
fac :=func(n);
       if isnat(n) then
                  if n=0 then return 1;
                  else return fac(n-1) * n ;
                   end;
       else print "no pertenece al dominio";
       end;
end;
```

2.2 NÚMERO DE ARREGLOS

Es frecuente en estadística obtener conclusiones sobre un conjunto de un número grande de n objetos, analizando entre ellos una muestra de i elementos. Esta muestra es con frecuencia, pequeña con respecto al conjunto universo, por eso muchas veces interesan las muestras ordenadas o grupos ordenados.

Estos grupos se llaman variaciones o arreglos i-arios por estar formados por i elementos u objetos (de los n dados) todos distintos y ordenados.

$$A_3^5 = A_2^5(5-2)$$

A_2^5

2			
ab	abc	abd	abe
ac	acb	acd	ace
ad	ade	adb	adc
.	.		
.		A_3^5	
.			
de	dea	deb	dec

- Se haría en clase un cuadro para un caso genérico
- Podemos deducir que todos los grupos del cuadro son arreglos de n elementos tomados de a i. En efecto, son grupos de i-1+1=i tomados de los n, ordenados y sin repetir, porque los i-1 iniciales constituían un arreglo y el elemento siguiente fue elegido entre los restantes del conjunto.
- No hay en el cuadro arreglos repetidos. En efecto, si están en la misma fila, difieren en el último elemento y si están en distintas filas tienen distinto grupo inicial.
- Cualquier arreglo de n elementos tomados de a i está en el cuadro por lo tanto ese cuadro es el conjunto de todos los arreglos de n elementos tomados de i en i. Su número es igual al número de filas por el de columnas.

$$A_i^n = A_{i-1}^n\{n-(i-1)\} \quad \forall i \geq 1$$

si n=0 definimos $A_0^n = 1$ o sea:

Definición:

$$\mathbf{ar : NxN \to N}$$

$$\mathbf{ar(n,\ i) = \begin{cases} 1 & \text{si } i = 0,\ n \geq i \\ ar(n,\ i-1) \times (n-i+1) & \forall i > 0 \end{cases}}$$

Programa en ISETL:

```
$arreglos
ar := func (n, i);
   if isnat(n) and isnat(i) then
                              if i=0 then return 1;
                              else return ar(n, i-1)*(n-
i+1);
                              end;
    end;
end;
```

Aplicado sucesivamente la definición recursiva podríamos calcular de otra manera el número de arreglos:

$$A_0^n = 1$$
$$A_1^n = A_0^n \, n$$
$$A_2^n = A_1^n (n-1)$$
$$A_3^n = A_2^n (n-2)$$
$$\vdots$$
$$A_i^n = A_{i-1}^n (n-(i-1))$$

$$A_i^n = n(n-1)(n-2)\ldots(n-i+1) \qquad \forall n \geq 1$$

También se podría llegar a lo mismo así:

$A = \{ a, b, c, .. \, j \} \qquad \# A = n$

Los arreglos de n elementos tomados de a uno son :

$a, \quad b, \quad c, \cdots \quad j \quad \Rightarrow \quad A_1^n = n$

$$a\begin{cases} b \\ c \\ d \\ . \\ . \\ j \end{cases} \qquad b\begin{cases} a \\ c \\ d \\ . \\ . \\ j \end{cases} \qquad c\begin{cases} a \\ b \\ d \\ . \\ . \\ j \end{cases} \qquad \cdots \qquad j\begin{cases} a \\ b \\ c \\ . \\ . \\ i \end{cases} \qquad A_2^n = n(n-1)$$

Y así llegamos a:

$$A_i^n = n\,(n-1)(n-2).....[n-(i-1)] \quad \forall n > 1$$

y

$$A_0^n = 1$$

2.3.1 NÚMERO DE PERMUTACIONES

Si i = n, los grupos ordenados se llaman permutaciones de orden n.
Dos permutaciones difieren solamente por el orden de sus elementos, pues ambas contienen a todos los elementos.

```
$permutaciones
 per := func (n);
        if isnat(n) then
                if n=0 then return 1;
                else return per(n-)*n;
                end;
        end;
 end;
```

2.4 NÚMERO DE COMBINACIONES

Los grupos de i objetos tomados de entre n, suelen ser también muestras en bloque, es decir, sin un orden determinado. Se les llama combinaciones i-arias o de orden i de n objetos. Dos combinaciones serán diferentes si y sólo si difieren en algún objeto (que está en una y no en la otra) no importando en cambio el orden.

En forma similar a la como trabajamos en arreglos llegaríamos a:

$$C_i^n = C_{n-1}^n \frac{(n-i+1)}{i}$$

Definición:

$$\mathbf{comb : NxN \rightarrow N}$$

$$\mathbf{comb(n, i) = \begin{cases} 1 & si \quad i = 0, \quad n \geq i \\ comb(n, i-1) \times (n-i+1)/i & \forall i > 0 \end{cases}}$$

Programa en ISETL:

```
$combinaciones limitando a n natural
 comb := func (n, i);
            if  isnat(n) and isnat(i) then
                                    if i=0 then return 1;
                                    else return  comb(n, i-1)*(n-i+1)/(i);
                                    end;
            end;
 end;
```

Llegaríamos a que deduzcan el número de combinaciones

$$C_i^n = \frac{n(n-1)(n-2)...(n-i+1)}{i!}$$

y luego por factoriales

$$C_i^n = \frac{n(n-1)(n-2)\ldots(n-i+1)(n-i)!}{i!(n-i)!}$$

$$C_i^n = \frac{n!}{i!(n-i)!}$$

$$\mathbf{comb2}(n,i) = C_i^n = \frac{n!}{i!(n-i)!} = \frac{\mathbf{fac}(n)}{\mathbf{fac}(i)\mathbf{fac}(n-i)}$$

Realizamos un programa con lo obtenido:

```
$combinaciones limitando a n natural
comb2:= func (n, i);
        if  isnat(n) and isnat(i) then
                        if i=0 then return 1;
                        else return fac(n)/(fac(n-
i)*fac(i));
                        end;
        end;
end;
```

y aquí también podemos observar que hay distintos métodos para llegar a los mismos resultados.

Es importante que los alumnos comprueben por inducción completa que

comb(n)=comb2(n)

$$\frac{C_{i-1}^n}{i}(n-i+1) = \frac{n!}{(n-i)!i!}$$

3. MÁXIMO COMUN DIVISOR

ALGORITMO DE EUCLIDES

Los divisores comunes a dos números son los mismos que los del par formado por el menor de ellos y el resto de la división del mayor por el menor.

Dados dos números **a** y **b** cuyos divisores comunes queremos hallar y $\mathbf{r_1}$ al resto de la división de **a** por **b**

$\mathbf{a=bq_1+r_1}$ siendo el resto $\mathbf{r_1<b}$

Para hallar los divisores comunes de **b** y $\mathbf{r_1}$ dividimos **b** entre $\mathbf{r_1}$, y así continuamos hasta hallar un resto nulo

$\mathbf{b= r_1q_2+r_2}$ $\mathbf{r_2<r_1}$

$\mathbf{r_1=r_2q_3+r_3}$ $\mathbf{r_3<r_2}$

.....

$\mathbf{r_{n-1}=r_n\ q_{n+1}+0}$ $\mathbf{0<r_n}$

La aplicación reiterada de esta propiedad conduce al conjunto de operaciones llamado algoritmo de Euclides o del máximo común divisor.

Los divisores comunes al último par: $\mathbf{r_n, 0}$ son los de $\mathbf{r_n}$, ya que 0 es divisible por cualquier número. Como los restos sucesivos son números naturales que cumplen las condiciones $\mathbf{b>r_1>r_2>r_3>}$**.....** debe llegarse necesariamente a un resto cero, $\mathbf{r_n}$ es el mayor divisor común del par **a, b**, razón por cual se le llama máximo común divisor o M.C.D. de **a** y **b**.

Definición:

$$gcd : N \times N \rightarrow N$$

$$gcd(x, y) = \begin{cases} y & \text{si } x \bmod y = 0 \\ gcd(y, x \bmod y) & \text{si } x \bmod y \neq 0 \end{cases}$$

Programa en ISETL:

```
$ gcd     "máximo común divisor"
gcd :=func(x, y);
        if (is_integer(x) and is_integer(y)) then
                if x mod y = 0 then return y;
                else return gcd(y , x mod y);
                 end;
        end;
end;
```

El algoritmo de Euclides es un algoritmo recursivo, o sea tenemos un caso base que el resto sea igual a cero y un término general que indica que se realicen todas las divisiones sucesivas hasta llegar al resto 0.

4. SUCESIONES DEFINIDAS POR RECURRENCIA

Podemos definir gran cantidad de sucesiones por recurrencia, indicando siempre el dominio y el procedimiento mediante el cual se pueden obtener todos los términos o sea su "término general" y el "caso base" que es indispensable para que quede bien definida.

4.1 SUCESIÓN DE FIBONACCI

La sucesión de Fibonacci, que se usa desde el siglo XIII, posee sorprendentes propiedades, en biología, por ejemplo, se ha determinado que el número de florcitas que forman el girasol es un número de la sucesión de Fibonacci; también las disposición de las ramas y las hojas de algunos árboles, así como el número de segmentos de la superficie de la piña.
Desde el punto de vista del arte la sucesión ha sido usada por Virgilio y otros poetas romanos que han escrito poemas en que la métrica está definida en términos de la sucesión de Fibonacci. También se ha podido establecer una relación con el diseño estético que los griegos utilizaron para construir sus edificios, la fachada del Partenón responde a la proporción áurea, que está relacionada con la sucesión de Fibonacci..-

Se llama sucesión de Fibonacci a la siguiente:

1, 1, 2, 3, 5, 8, 13, 21,.....

vemos que partiendo de los números 1 y 1, cada término es la suma de los dos precedentes.

Definición:

$$f : N^* \to N$$

$$f : f(n) = \begin{cases} 1 & \text{si} \quad n = 1 \\ 1 & \text{si} \quad n = 2 \\ f(n-2) + f(n-1) & \text{si} \quad n > 2 \end{cases}$$

Programa en ISETL

```
fib :=func(n);
        if (isnat(n) and n>0) then
                                    if n=1 then return 1;
                                     end;
                                    if n=2 then return 1;
                                    else return fib(n-2) + fib(n-1) ;
                                     end;
        end;
end;
```

Aclaración

Notar que a diferencia de los ejemplos anteriores, f(n) depende de dos términos anteriores, f(n-1) y f(n-2) por lo tanto en este caso es necesario poner dos casos bases.

5. ANEXO

Ahora vamos a trabajar operaciones con números naturales que pondremos en el programa ISETL como funciones predefinidas para que puedan ser utilizadas en aquellos ejemplos que lo requieran o que puedan ser estudiadas con mayor profundidad según el tema a tratar.

OPERACIONES CON NÚMEROS NATURALES

Se define sucesor y predecesor de n, para que faciliten la creación de programas de acuerdo a la definición de las operaciones.

```
$sucesor
suc :=func(n);
        if isnat(n)
        then return n+1;
        end;
end;
```

```
$predecesor
pred :=func(n);
          if isnat(n) and n>0
          then return n-1;
          end;
end;
```

ADICIÓN

Se define la función adición de dos números naturales, utilizando las funciones ya definidas, sucesor y predecesor de forma tal que:

suma(x, y - 1) + 1

nos queda***:***

suc(suma(x, pred(y)))

Definición:

$$suma : N \times N \rightarrow N$$

$$suma(x, y) = \begin{cases} x & \text{si } y = 0 \\ suc(suma(x, pred(y))) & \text{si } y \neq 0 \end{cases}$$

Programa en ISETL

```
$suma:(NxN)->N
suma := func(x,y);
        if isnat(x) and isnat(y) then
                                if y=0 then return x;
                                else return
suc(suma(x,pred(y)));
                                end;
        end;
end;
```

MULTIPLICACIÓN

Se define la función multiplicación de dos números naturales usando la función predefinida predecesor de forma tal que:

suma(producto(x, y - 1), x)

nos queda***:***

suma(producto(x, pred(y)), x)

Definición:

$$producto : N \times N \rightarrow N$$

$$producto(x, y) = \begin{cases} 0 & \text{si } y = 0 \\ suma(producto(x, pred(y)), x) & \text{si } y \neq 0 \end{cases}$$

Programa en ISETL:

```
producto :=func(x, y);
            if isnat(x) and isnat(y) then
                                    if y=0 then return 0;
                                    else return suma(producto(x , pred(y)),x);
                                     end;
            end;
end;
```

POTENCIACIÓN DE EXPONENTE NATURAL

Se define la función potenciación de dos números naturales usando la función predefinida predecesor de forma tal que:

pot(x, y - 1)*x

utilizamos***:***

producto(pot(x, pred(y)), x)

Definición:

$$pot : N \times N \rightarrow N$$

$$pot(x, y) = \begin{cases} 1 & \text{si } y = 0,\ x \neq 0 \\ producto(pot(x, pred(y)), x) & \text{si } y \neq 0 \end{cases}$$

Programa en ISETL:

```
pot :=func(x, y);
        if isnat(x) and isnat(y) then
                                    if y=0 and x/=0 then return 1;
                                   else return producto(pot(x , pred(y)),x);
                                    end;
        end;
end;
```

COMENTARIO

Se probó el programa en una clase práctica de 5º año de un grupo de alumnos con conocimientos de inglés y manejo de PC. Se utilizó la herramienta computacional para crear los programas y realizar pruebas sucesivas. La rapidez de respuesta numérica de la herramienta computacional permitió un juego de búsqueda-acierto-error desarrollado por los alumnos, dificultoso de realizarse a ese nivel por medios manuales.

Para poder realizar los programas se buscaron problemas simples, pero que promovieron un intenso trabajo de reflexión.

El esquema metodológico se basó en la participación activa de los estudiantes usando el recurso de la discusión. Las actividades se organizaron con propuestas sucesivas emergentes de un planteo inicial utilizado como detonador y por las preguntas de los propios alumnos.

Es aquí donde se hace evidente la importancia de la computadora para la construcción del conocimiento mediante razonamientos plausibles, intuitivos al comienzo, con la realización de pruebas que estimulan la integración de conceptos. En todo momento los docentes trataron de estimular los razonamientos a los efectos de llegar a la resolución de los problemas.

BIBLIOGRAFÍA

Mercedes Anido de López - Héctor E. Rubio Scola ***"Un ejemplo de aprendizaje en el sentido de Polya"***
Revista Latinoamericana de Investigación en Matemática Educativa
Vol. 1, Núm. 3, noviembre, 1998

Guillermo Espinosa ***"La sucesión de Fibonacci"***

Revista Matemática Mexicana
2ª serie, Num 5, noviembre de 1989

K. Ribnikob ***"Historia de las Matemáticas"***
Editorial Mir, Moscú, 1987

Margaret Willerding ***" Conceptos matemáticos un enfoque histórico"***
Serie Complementaria de Matemáticas C.E.C.S.A., diciembre de 1969

William E. Fenton –Ed Dubinsky ***"Introduction to Discrete Mathematics with ISETL"***
Editorial Springer, 1996

Edith Gallo-Stelio-Haniotis_Julio Silvera ***"Mikrakys 5º Año Tomo 1"***
Editorial Fin de Siglo. Uruguay 1998

Winbfried Grassmann-Jean-Paul Tremblay ***"Matemática Discreta y Lógica"***
Editorial Prentice Hall

Luis Osín ***"Introducción al Análisis Matemático"***
Editorial Kapeluz S.A. Uruguay 1966

Richard Courant-Herbert Robbins ***"¿Qué es la matemática?"***
EDITORIAL Aguilar S.A. Uruguay1967

Rey Pastor ***"Elementos de análisis algebraico"***
EDITORIAL Macagno y Cia.Argentina1966